Je débute en BASIC



C. DELANNOY

EYROLLES

AMSTRAD

JE DÉBUTE EN BASIC AMSTRAD

CHEZ LE MÊME ÉDITEUR

Du même auteur

DELANNOY - *Apprendre à programmer en BASIC.*

- *Initiation à la programmation.*

- *Initiation à Multiplan avec exercices et corrigés.*

- *Les fichiers en BASIC sur micro-ordinateur.*

Autres ouvrages

LEPAPE - *L'assembleur facile du Z 80.*

DAX - *CP/M et sa famille. Guide d'utilisation.*

PEZERET - *Créez vos jeux d'aventure sur micro-ordinateur. Méthodes et idées.*

REVERCHON et DUCAMP - *Mathématiques sur ordinateur.*
1. Analyse.
2. Algèbre.

O' MALLEY - *25 programmes graphiques en BASIC Microsoft.*

DELAHAYE - *Dessins géométriques et artistiques avec votre micro-ordinateur.*

WEIDENFELD, MATHIEU et PEROLAT - *Logo.*

WEIDENFELD, BRUILLARD et PEROLAT - *Aller plus loin en Logo.*

JE DÉBUTE EN BASIC AMSTRAD

par

Claude DELANNOY



61, boulevard Saint-Germain — 75005 Paris
1985

Si vous désirez être tenu au courant de nos publications, il vous suffit d'adresser votre carte de visite au :

Service « Presse », Éditions EYROLLES
61, Boulevard Saint-Germain,
75240 PARIS CEDEX 05,

en précisant les domaines qui vous intéressent. Vous recevrez régulièrement un avis de parution des nouveautés en vente chez votre libraire habituel.

Avant-propos

Nous avons conçu ce livre pour qu'il soit le premier compagnon de route de nombreux apprentis-programmeurs (jeunes ou moins jeunes...). Dans ce but, nous nous sommes limités à un langage très simple et nous ne supposons aucune connaissance préalable de quelque sorte que ce soit.

Partant du principe que la programmation ne peut s'apprendre que par la pratique, nous avons adopté une pédagogie active associée à une démarche très progressive. Chaque notion est présentée sur un exemple simple qui peut être immédiatement essayé sur votre micro-ordinateur. Mais, en outre, vous êtes invité à expérimenter des situations très variées repérées par le titre « Faites-vous la main ». C'est de la pratique de ces exercices (on pourrait presque dire de ces « gammes ») que dépend votre réussite. Ne les négligez donc surtout pas !

Très souvent, le débutant se trouve arrêté par un comportement imprévu de l'ordinateur. C'est pour cette raison que, dès le début, nous vous familiarisons avec les erreurs qu'il est possible de commettre. Nous vous montrons à la fois comment elles se manifestent et comment y remédier. En procédant ainsi, la situation d'erreur perdra son aspect « bloquant ».

A la fin des premiers chapitres, vous trouverez des exercices qui vous incitent à appliquer les notions fraîchement acquises. Ne cédez pas trop vite à la tentation de « pianoter » sur votre micro. Prenez le temps nécessaire à la réflexion, source certaine de progrès.

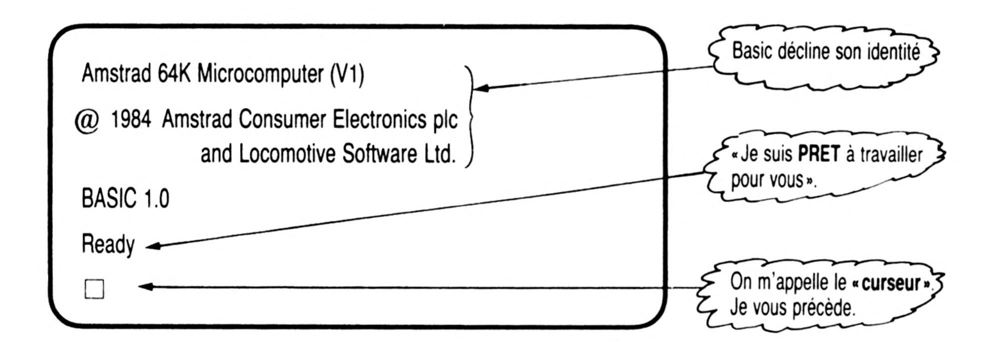
Ce livre est utilisable aussi bien avec l'Amstrad CPC464 (unité centrale + cassette intégrée) qu'avec l'Amstrad CPC664 (unité centrale + disquette intégrée).

Table des matières

I. Quand Basic vous obéit immédiatement

Contact !

Lorsque vous allumez votre Amstrad, vous voyez apparaître à l'écran quelque chose comme ceci :



Le mot **« Ready »** (mot anglais signifiant « prêt » vous montre que Basic attend que vous lui fournissiez des instructions. C'est ce que vous allez faire en utilisant le clavier.

Le carré clignotant, qu'on appelle « curseur » vous montre où va s'afficher ce que vous allez taper.

Première instruction

Pour commencer, nous vous proposons de taper :

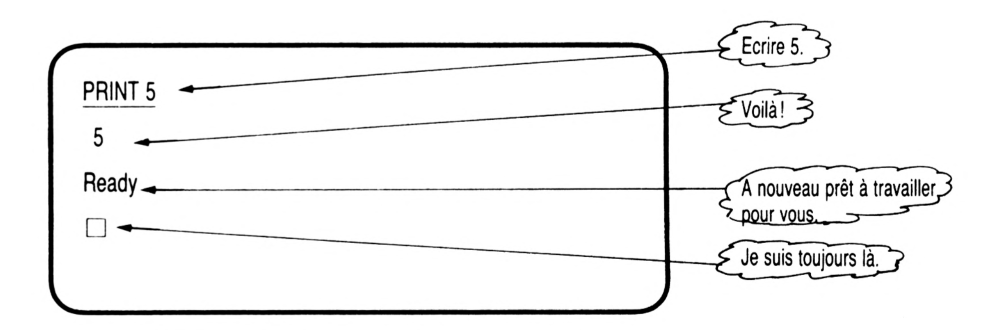
PRINT 5

Pour cela, vous frappez successivement sur les touches marquées P, R, I, N, T puis sur la barre d'espace puis sur 5. (Vous pouvez écrire aussi bien en majuscules qu'en minuscules.) A chaque fois, le curseur se déplace sur la position suivante.

Mais, direz-vous, rien ne se passe ! Cela est normal, car il vous faut maintenant dire à Basic que vous avez fini de taper votre intruction. Pour cela, vous appuyez sur la touche :



Basic exécute alors ce que vous lui avez demandé. Vous voyez apparaître à l'écran, à la suite de ce qu'il y avait déjà (que nous n'avons pas reproduit ici) :



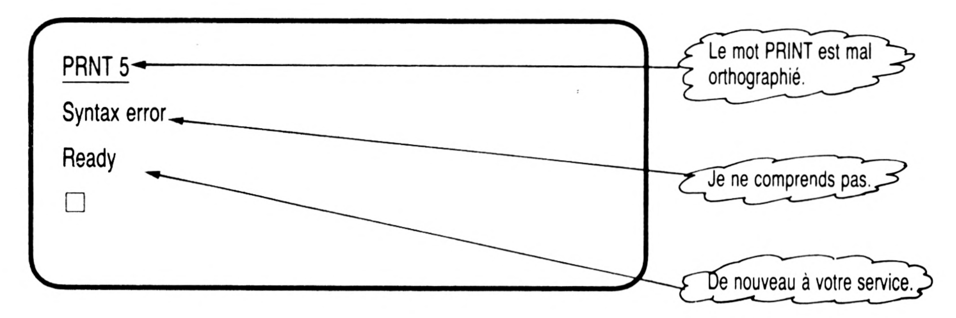
Pour que ce que vous tapez se distingue de ce que vous affiche Basic en réponse, nous l'avons souligné. Naturellement, sur votre écran, il n'y aura pas de différence.

PRINT est un mot connu de Basic. Il signifie **Ecrire**. Ici, vous lui avez donc demandé d'écrire 5. C'est ce qu'il a fait immédiatement sur la ligne suivante. Il vous signale ensuite qu'il est à nouveau prêt à recevoir une instruction en écrivant Ready.

En cas d'erreur

Que va-t-il se passer si vous faites une erreur en tapant votre instruction?

Si vous ne vous apercevez pas de votre faute avant d'avoir tapé « ENTER », il est probable que Basic ne comprendra pas ce que vous lui demandez. Par exemple, si vous faites une faute d'orthographe en tapant le mot PRINT, voici ce qui se passera :



La réponse « Syntax error » que vous obtenez signifie « erreur de syntaxe ». Autrement dit, Basic vous dit que quelque chose est mal orthographié et qu'il ne peut pas exécuter ce que vous lui demandez. Cela ne l'empêche pas d'attendre une nouvelle instruction en vous affichant Ready.



Faites-vous la main

* Tapez les instructions suivantes, sans chercher à corriger celles qui comportent une erreur. Essayez de prévoir la réponse que vous fournira Basic dès que vous appuyerez sur ENTER.

```
PRIMT 25
PRINT 1995
ECRIS 35
PRINT - 8
BONJOUR
POURQUOI
```

Lorsque vous pouvez corriger votre erreur

Et si vous découvrez votre erreur avant d'avoir tapé RETURN !

Dans ce cas, rassurez-vous, il vous est facile de la corriger. Il vous suffit de revenir « en arrière » en appuyant sur la touche marquée :





A chaque fois que vous pressez cette touche, le curseur recule d'une position et le dernier caractère est effacé. Il vous suffit ensuite de taper les bonnes touches pour obtenir l'instruction correcte.

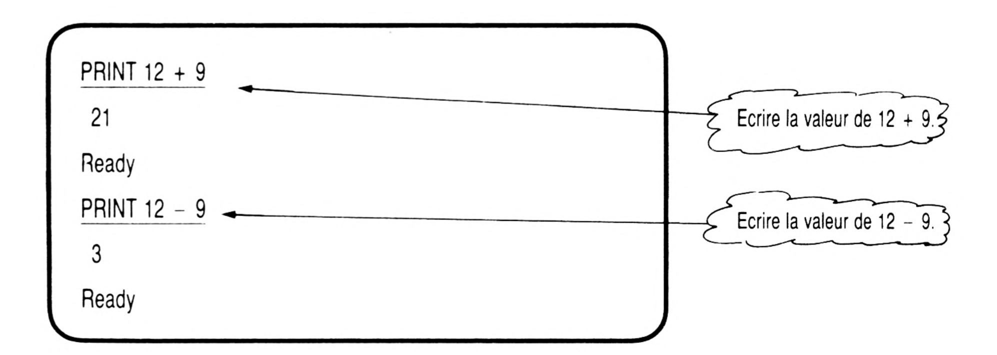


Faites-vous la main

- Commencez de taper: PRIT. Revenez en arrière et corrigez de façon à obtenir PRINT 8. Tapez ENTER pour faire exécuter.
- Commencez de taper: PRNT 834 puis revenez en arrière autant de fois que nécessaire pour pouvoir corriger et obtenir PRINT 834.
- Entraînez-vous à corrigier ainsi n'importe quelle faute de votre choix, jusqu'à ce que cela vous paraisse tout à fait naturel.

Un peu de calcul

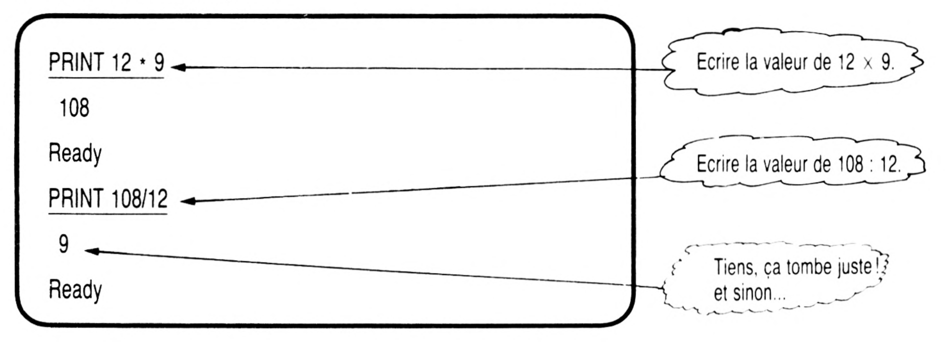
Vous pouvez demander à Basic de vous fournir le résultat d'une opération. Voici des exemples d'additions et de soustractions.



La multiplication et la division sont tout aussi faciles à employer. Il suffit de savoir que :

- la multiplication se note *
- la division se note /

En voici des exemples :





Faites-vous la main

- Essayez ces instructions :
  PRINT 100 * 12
  PRINT 25 * 182
  PRINT 750/25
- Vous pouvez grouper plusieurs opérations ; essayez :
  PRINT 5 + 3 + 2
  PRINT 12 + 8 − 10
  PRINT 8 * 8 * 10
  PRINT 9 * 8/3
- Essayez des instructions de votre choix.

Notre première variable

Si vous tapez l'instruction :

A = 5 (sans oublier ENTER)

vous obtenez :

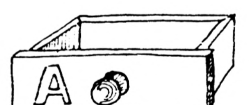
```
A = 5
Ready
```

Apparemment Basic n'a rien fait. Cependant, votre instruction ne lui a pas déplu puisqu'il n'a pas écrit de « message d'erreur ». Le mot « Ready » montre qu'il est prêt à exécuter une nouvelle instruction.

Basic n'a-t-il vraiment rien fait? Proposons-lui l'instruction : PRINT A.

```
A = 5
Ready
PRINT A
 5
Ready
```

Voyons de près ce qui s'est passé depuis le début (A = 5).

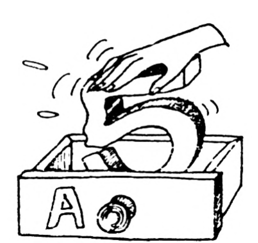


* *L'instruction :*

A = 5

signifie :

- Choisir un emplacement de la mémoire et lui donner le nom A.
- Y ranger la valeur 5



* *L'instruction :*

PRINT A

signifie : écrire quelle est la valeur contenue dans A.

En jargon informatique, on dit que A est une VARIABLE.

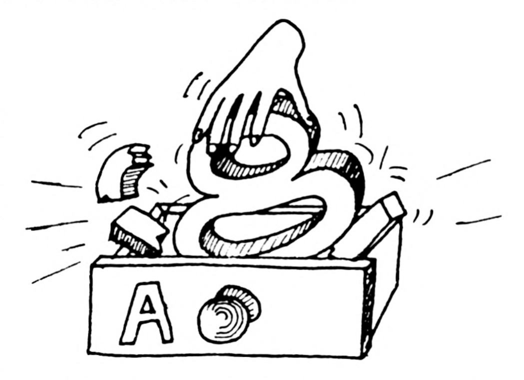
On dit aussi que A = 5 est une instruction *« d'affectation »* : elle *« affecte »* à la variable A la valeur 5.

Et quand la variable change de valeur

Que va faire Basic si nous lui demandons d'exécuter l'instruction :

```
A = 8
```

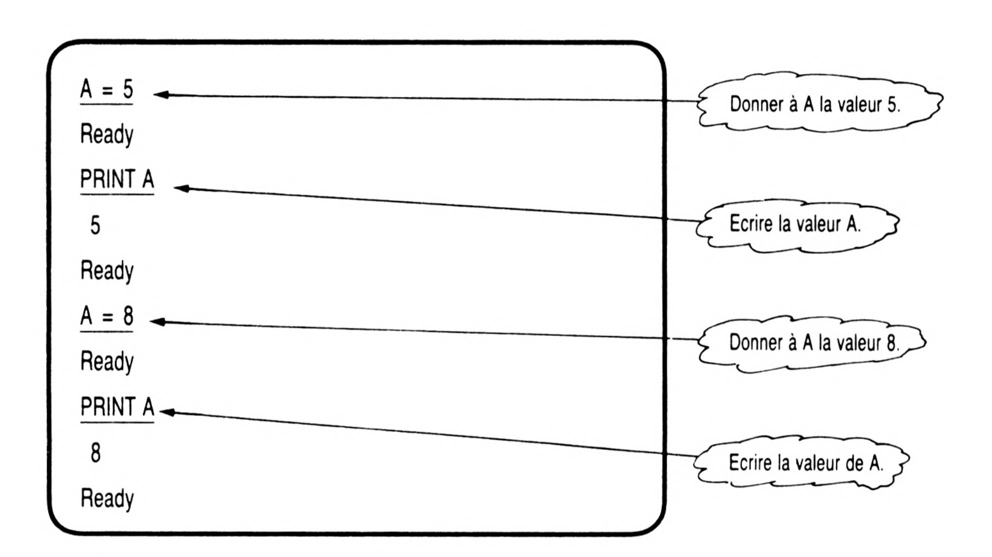
Cette fois l'emplacement A existe déjà. Vous demandez à Basic d'y placer la valeur 8. C'est bien ce qu'il va faire, en détruisant du même coup la valeur 5 qui s'y trouvait auparavant.



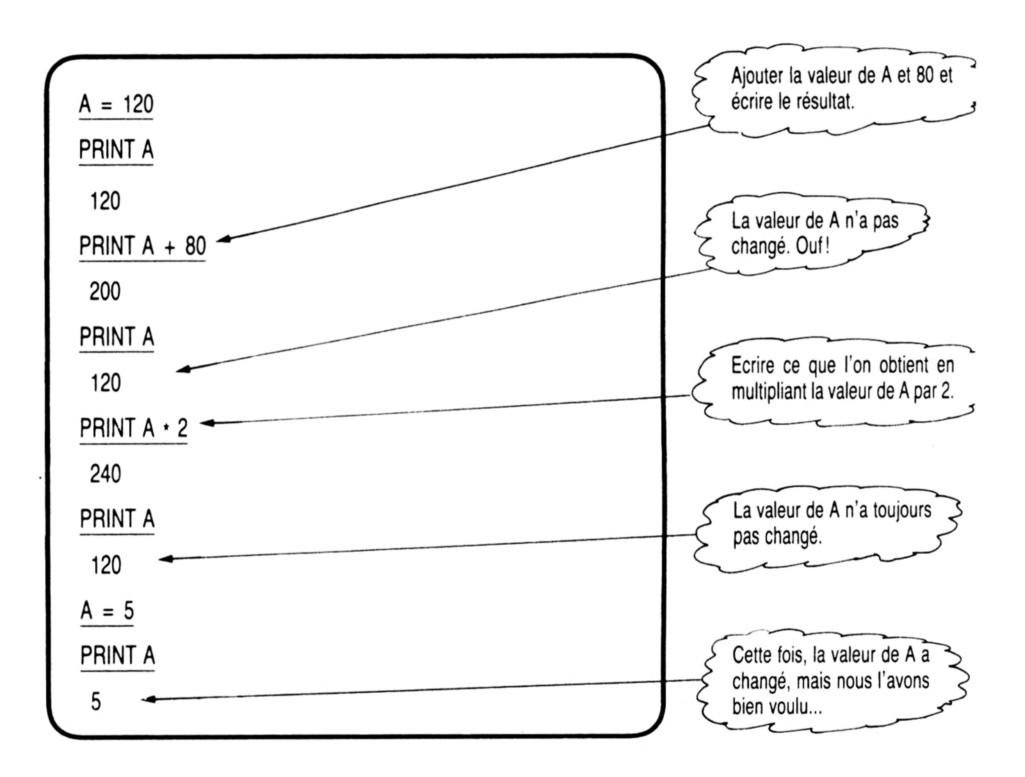
Pour vous en convaincre, il vous suffit de demander à Basic de vous écrire la valeur contenue dans A par :

```
PRINT A
```

Pour nous résumer, voici ce qui doit apparaître sur votre écran :



Voici un autre exemple qui vous montre d'autres façons d'utiliser une variable. Cette fois, pour alléger un peu notre texte, nous avons supprimé les mots «Ready» qui apparaissent après l'exécution de chaque instruction.



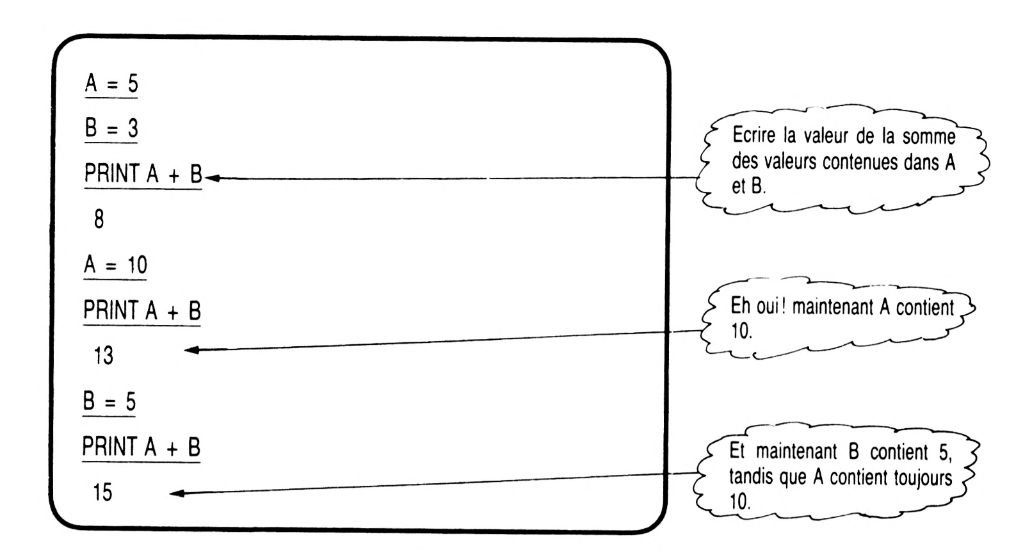
Faites-vous la main

Exécutez successivement ces instructions:

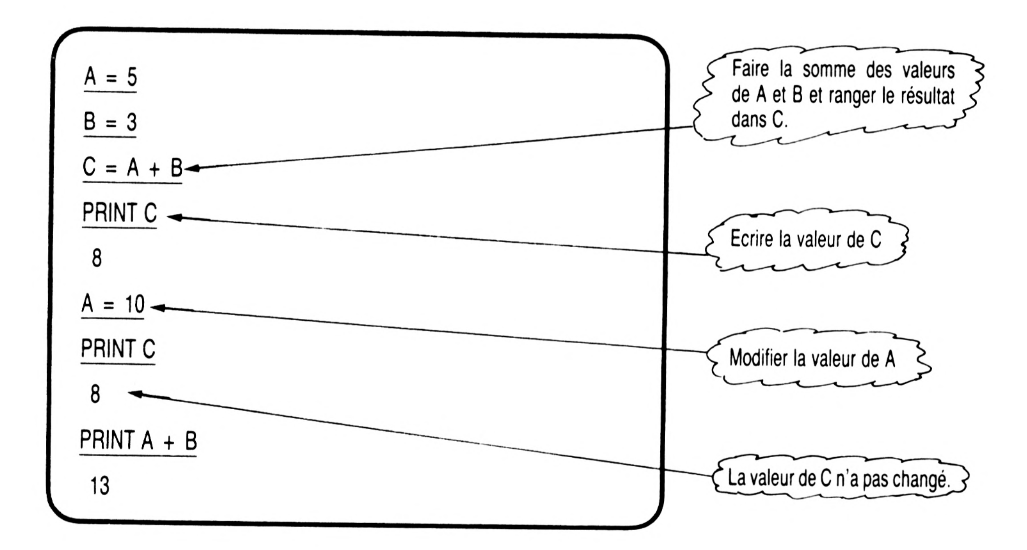
```
A = 12
PRINT A + A
PRINT 2 * A
PRINT A * A
PRINT A/A
PRINT A - A
PRINT A * 12
PRINT A
A = 15
PRINT A * 12
PRINT A * A
PRINT A
```

Avec plusieurs variables

Un premier exemple :



Et un second :



Faites-vous la main

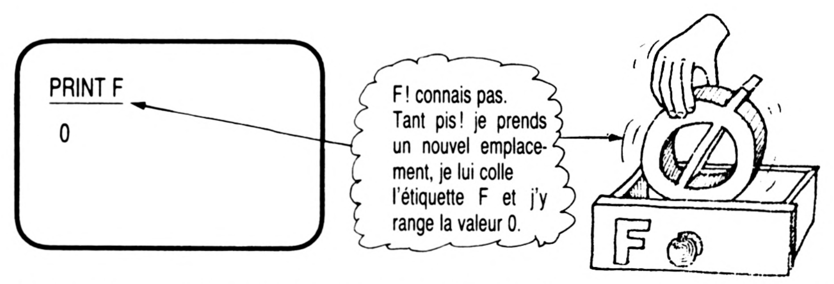
Exécutez successivement les instructions suivantes :

```
A = 9
B = 18
S = A + B
PRINT S
D = B - A
PRINT D
D = A - B
PRINT D
P = A * B
PRINT P
Q = B/A
PRINT Q
```

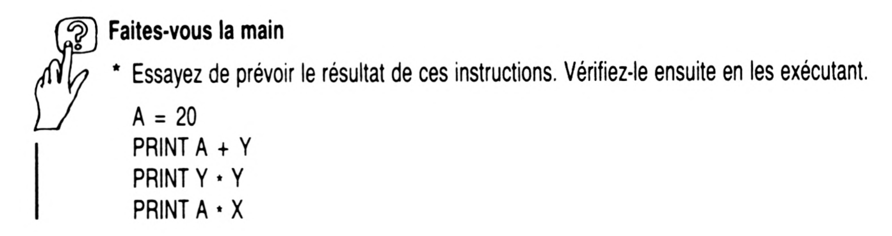
Quand une variable n'a pas de valeur

Que va-t-il se passer si vous utilisez une variable à laquelle vous n'avez jamais attribué de valeur ?

Pour le savoir, demandez à Basic de vous dire ce que contient F (nous n'avons jamais utilisé ce nom jusqu'ici).



Rassurez-vous ! Le zéro que vous obtenez n'est pas une (mauvaise) note que vous décerne Basic. Il s'agit tout simplement de la valeur de F.



Faites-vous la main

* Essayez de prévoir le résultat de ces instructions. Vérifiez-le ensuite en les exécutant.

```
A = 20
PRINT A + Y
PRINT Y * Y
PRINT A * X
```

Pour effacer l'écran



Il est probable que vous avez déjà éprouvé l'envie d'effacer l'écran, peut-être pour faire disparaître quelques sottises accompagnées de «Syntax error».

Bien sûr, vous pouvez toujours y parvenir en «éteignant» votre ordinateur et en le rallumant (après avoir attendu quelques secondes). Mais vous obtenez alors à nouveau les lignes de présentation de Basic. D'autre part, tout a alors disparu, en particulier les variables et leurs valeurs.

Une solution plus douce consiste à faire exécuter l'instruction d'effacement d'écran :

```
CLS
```

Essayez cette instruction et notez que le mot Ready réapparaît en haut de l'écran.

Il est très important de savoir que cette instruction se contente uniquement d'effacer l'écran. Elle ne modifie rien de ce que Basic a pu mettre en mémoire.

En particulier, les valeur des variables ne sont pas atteintes.

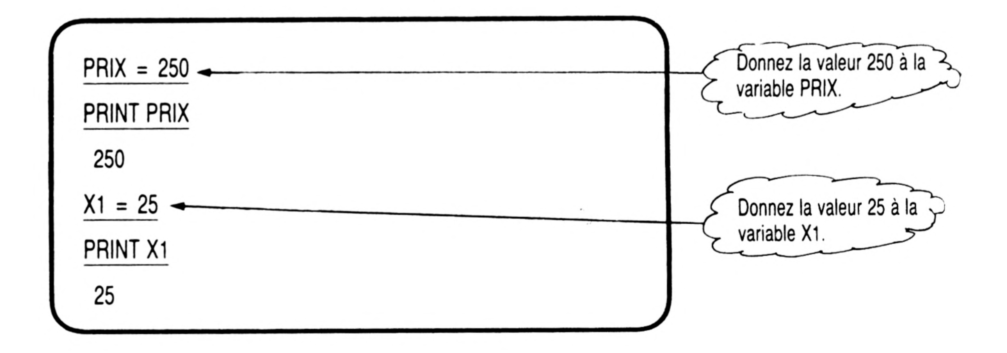
Faites-vous la main

- Exécutez ces instructions
  A = 15
  B = 20
- Effacez l'écran.
- Examinez le contenu des variables A et B en exécutant :
  PRINT A
  PRINT B

Des noms de variables plus parlants ! Oui, mais...

Jusqu'ici, nous avons choisi des noms de variables formés d'une seule lettre : A, C, X, Y, etc.

En fait, Basic vous permet d'utiliser des noms comportant plusieurs lettres ; les chiffres sont même autorisés à condition de ne pas apparaître au début du nom. Essayez :



Bien pratique, direz-vous ! Nous pouvons donc choisir des noms de variables adaptés à ce qu'elles représentent, par exemple : MONTANT, POSITION, AGE, etc.

Toutefois, essayez :

```
NOT = 16
Syntax error
FOR = 20
Syntax error
TEST = 25
Syntax error
```

Les noms NOT, FOR et TEST semblent ne pas convenir à Basic. Pourquoi ? Il existe en fait un certain nombre de mots qui possèdent une signification précise pour Basic. On les appelle *« mots-clés »*. Pour l'instant, nous n'en avons rencontré qu'un seul : PRINT. Mais il en existe beaucoup d'autres : INPUT, FOR, NEXT, NOT, ON, OR, TEST, etc. Et Basic n'accepte pas qu'un tel mot soit employé comme nom de variable.

Mais, direz-vous, comment connaître tous les mots-clés ? En fait, vous rencontrerez peu de problèmes si vous vous limitez à des noms de deux caractères. En effet, les seuls mots-clés de deux lettres employés par votre Amstrad sont :

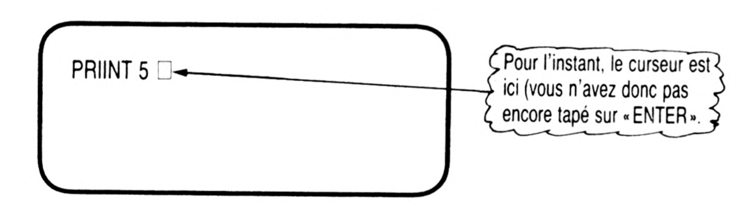
IF, ON, OR, TO, PI, SQ, FN et EI.

Pour faciliter vos corrections

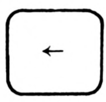
Nous avons vu comment il était possible de corriger une instruction, à l'aide de la touche « DEL » (tant que vous n'avez pas appuyé sur « ENTER »).

En fait, il existe d'autres touches vous permettant d'effectuer des corrections. Voyez ces deux exemples :

Exemple 1 : si vous tapez :



vous pouvez faire reculer le curseur sur l'une des deux lettres I, à l'aide de la touche :



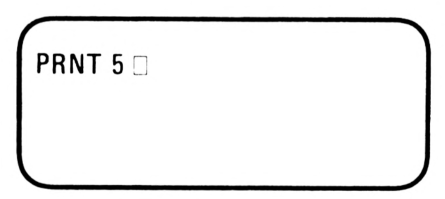
Cette fois, rien ne s'est effacé. Si maintenant vous appuyez sur



vous voyez disparaître le I sur lequel se trouvait le curseur.

Maintenant que votre instruction est correcte, vous pouvez l'exécuter en tapant « ENTER ». Il n'est pas nécessaire de ramener le curseur à droite du 5.

Exemple 2 : si vous avez tapé :



vous pouvez faire reculer le curseur sur la lettre N. Ensuite vous tapez I qui vient s'insérer à gauche du curseur, entre R et N.

Maintenant, votre instruction est correcte (le curseur est sur le R). Vous pouvez l'exécuter en tapant « ENTER ».



Faites-vous la main

Habituez-vous à corriger n'importe quelle faute à l'aide des touches ←, →, DEL et CLR. N'oubliez pas que :

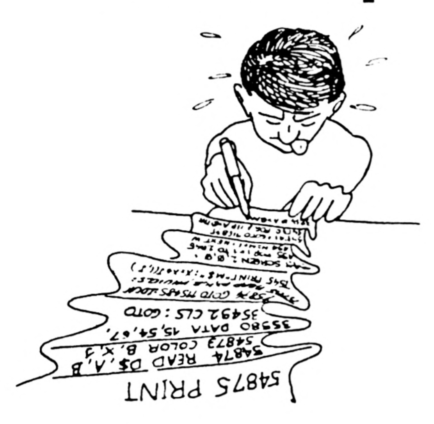
CLR efface le caractère situé **sous** le curseur
DEL efface le caractère situé **à gauche** du curseur.

EXERCICES (corrigés à la fin du livre) :

Nous vous conseillons de chercher d'abord à résoudre ces exercices « sur le papier ». Vous pourrez ensuite, si vous souhaitez, taper les instructions correspondantes sur votre Amstrad.

1. Que fait l'instruction : PRINT 15 + 3
2. Donnez une instruction permettant d'obtenir la valeur de 365 × 21.
3. Donnez une instruction permettant d'affecter la valeur 150 à une variable nommée E.
4. Que font ces instructions (lorsqu'on les exécute dans cet ordre).
   A = 5
   PRINT A * A
5. Même question avec :
   A = 20
   B = 5
   C = A * B
   PRINT C
6. Même question avec :
   A = 15
   PRINT A + 3
   PRINT A
   A = 8
   PRINT A * A
   PRINT A
7. Vrai ou faux ? Lorsque l'on efface l'écran, les variables conservent leur valeur.

II. Numéroter des instructions : tout un programme



Dans le précédent chapitre, nous avons appris à donner des instructions à Basic qui les exécutait immédiatement. Mais il n'en gardait aucune trace. La preuve, si nous voulions exécuter à nouveau une instruction, nous devions la retaper. Nous allons maintenant apprendre comment faire « mémoriser » des instructions ; cela constituera un « programme ». Avant d'aborder la suite, nous vous conseillons de *« repartir à zéro »*, en éteignant quelques instants votre Amstrad.

Notre premier programme

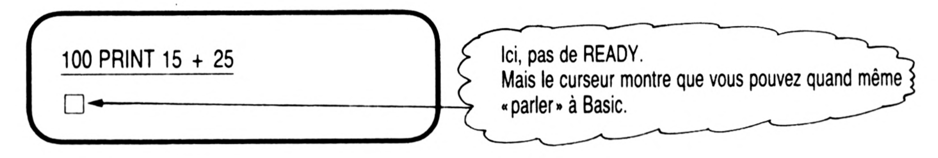
Vous savez que si vous tapez :

```
PRINT 15 + 25
```

vous obtenez immédiatement le résultat, suivi du mot « Ready » :

```
PRINT 15 + 25
 40
Ready
□
```

Mais si vous tapez cette même instruction précédée d'un numéro (par exemple 100), il n'en va plus de même :

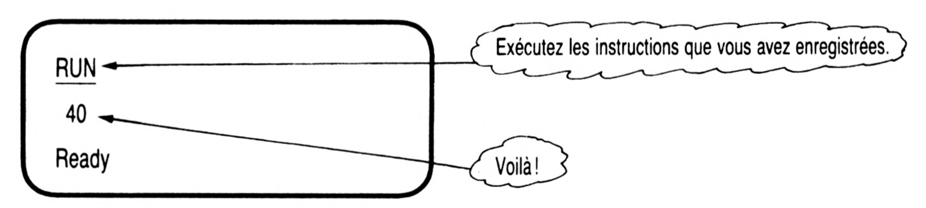


Apparemment, rien ne se passe. En fait, comme votre instruction comporte un numéro, Basic a compris qu'il devait l'**enregistrer** sans l'exécuter.

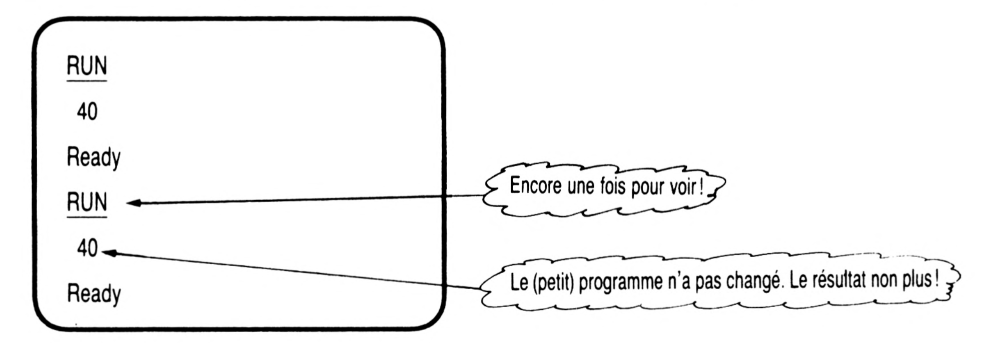
Vous pouvez ainsi enregistrer plusieurs instructions, ce qui constituera un « programme ». Pour l'instant, nous allons nous contenter de « jouer » un peu avec ce programme formé d'une seule instruction portant le numéro 100.

Pour « exécuter » un programme

Tapez : RUN (sans numéro et en n'oubliant pas de taper ensuite sur la touche « ENTER ») :



Mais, direz-vous, j'aurais obtenu le même résultat en tapant simplement l'instruction en question, sans l'enregistrer ! Pour ce petit exemple, c'est tout à fait vrai. Malgré tout, votre programme est toujours là et vous pouvez l'exécuter autant de fois que vous le voulez sans avoir à le taper de nouveau :



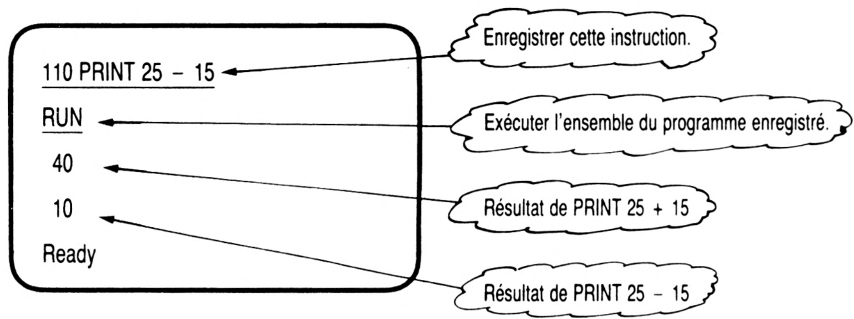
Bien sûr, tout cela sera beaucoup plus intéressant quand le programme sera plus gros. Patience...

Faisons grossir notre programme

Nous allons ajouter une nouvelle instruction à notre programme :

PRINT 25 - 15

Il nous suffit de la taper, en la faisant précéder d'un numéro, par exemple 110. Si nous tapons ensuite RUN, voici ce que nous obtenons :



Comment Basic sait-il dans quel ordre il doit exécuter vos instructions ? Il se sert tout simplement des numéros que vous leur attribuez. Ici, nous avions choisi 100 et 110. Basic a donc exécuté d'abord l'instruction 100, puis l'instruction 110.

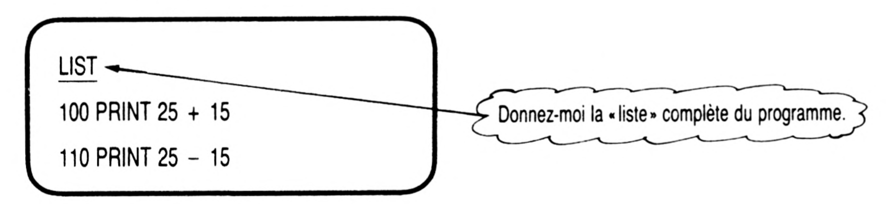
Ready or not Ready ?

Vous avez constaté qu'à chaque fois que vous enregistrez une nouvelle instruction, Basic se contente de vous présenter le curseur à la ligne suivante. Il n'affiche pas le mot « Ready ». Curieux, direz-vous ! En fait, cela est très pratique quand vous enregistrez plusieurs instructions de suite. Dans ce cas, vos lignes se suivent sans être entrecoupées de « Ready » qui seraient bien désagréables.

En pratique, donc, sachez que vous pouvez « parler » à Basic dès que le curseur apparaît à l'écran. Dans la suite de ce livre, nous ne mentionnerons plus le mot « Ready », sauf cas particulier.

Pour savoir ce que Basic a enregistré

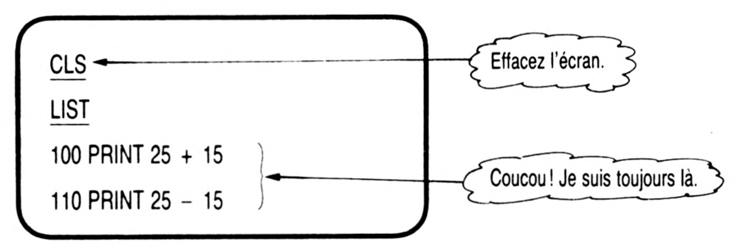
Tapez LIST (sans numéro) :



Ici, il n'y a pas de surprise car vous vous souvenez des instructions que vous avez tapées. Peut-être même sont-elles encore visibles sur l'écran. Toutefois, cela vous permet de connaître avec certitude ce que Basic a enregistré.

Pour effacer un programme

Nous avons déjà vu que lorsque vous effacez l'écran, les variables ne sont pas modifiées. De la même façon, le programme n'est pas détruit ; la preuve :

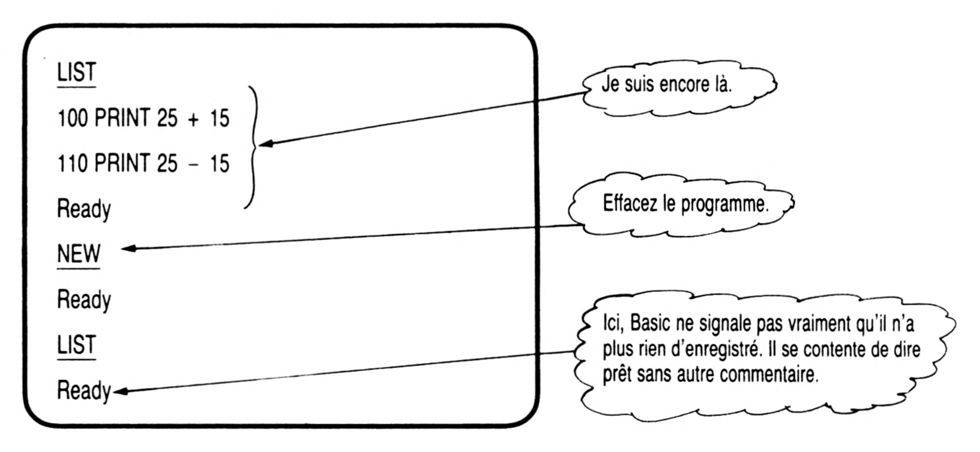


Pour effacer un programme, il faut employer l'instruction :

NEW

Ce mot anglais signifie « nouveau ». C'est une façon de dire que vous avez l'intention de préparer un nouveau programme et donc qu'il faut effacer l'ancien.

Expérimentons cette instruction :

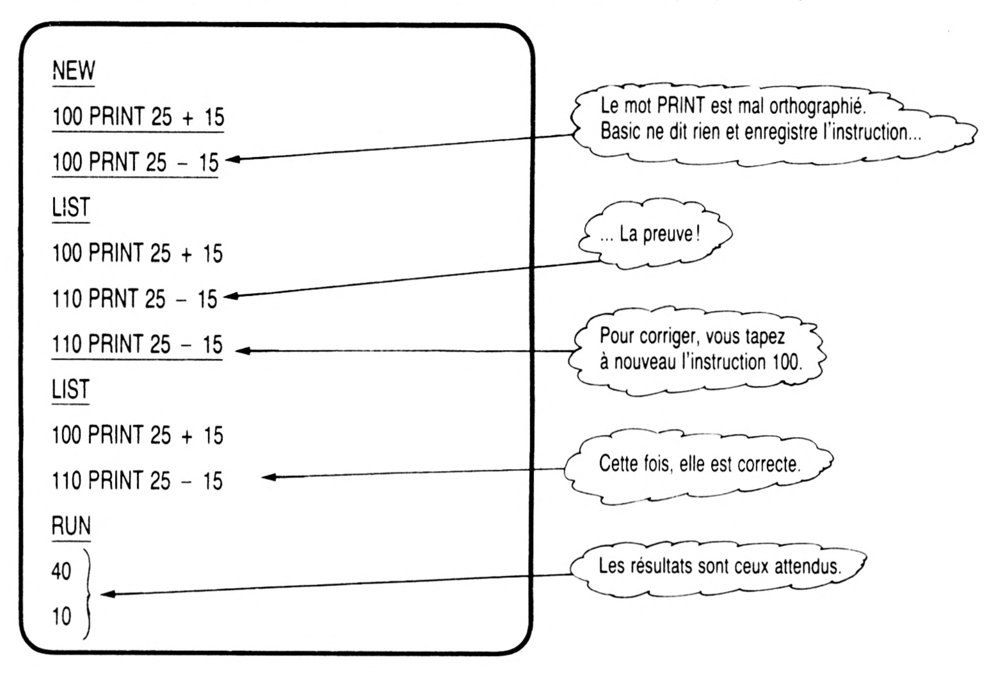


Ici, exceptionnellement, nous avons fait apparaître les « Ready ».

En cas d'erreur

Nous avons vu comment corriger une instruction que vous êtes en train de taper, tant que vous n'avez pas appuyé sur « ENTER ». Vous savez que vous pouvez utiliser pour cela les touches ←, →, DEL et CLR.

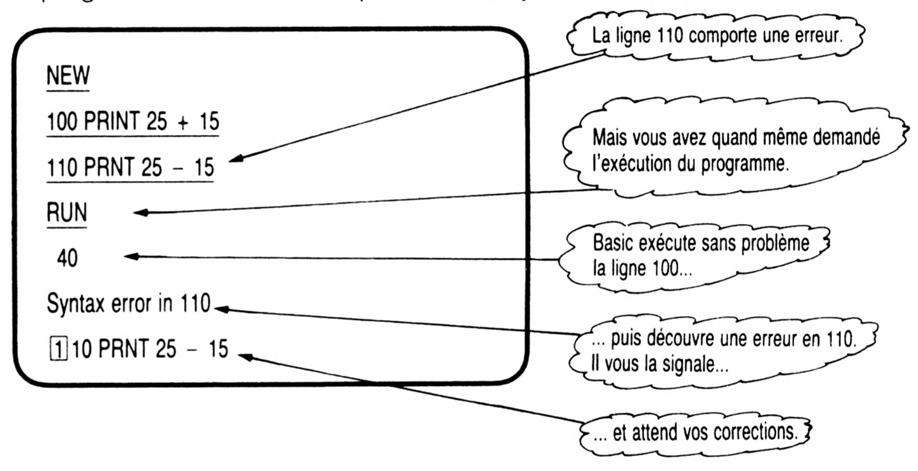
Mais comment faire si vous enregistrez une instruction comportant une erreur ? Essayez ceci (en tapant exactement ce que nous vous proposons) :



Vous voyez qu'il est facile de modifier une instruction : il suffit de la taper à nouveau.

Et quand vous ne voyez pas votre erreur !

Mais il se peut que vous ne vous aperceviez pas de votre erreur avant d'exécuter le programme. Que va-t-il se passer? Essayez à nouveau :



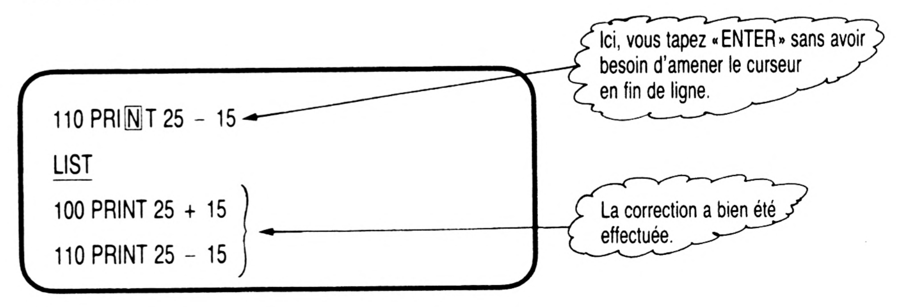
Vous voyez que Basic ne vous signale votre erreur qu'au moment où il cherche à exécuter l'instruction correspondante. Il vous affiche alors l'instruction coupable en plaçant le curseur en début de ligne.

Il vous suffit alors de faire vos corrections comme nous avons déjà appris à le faire avec les touches ←, →, DEL et CLR. Ici, par exemple, vous pouvez amener le curseur sur la lettre N :

110 PRNT 25 - 15

Vous tapez ensuite la lettre I qui vient s'insérer entre le R et le N. Comme l'instruction est maintenant correcte, il ne vous reste plus qu'à taper sur

« ENTER » :

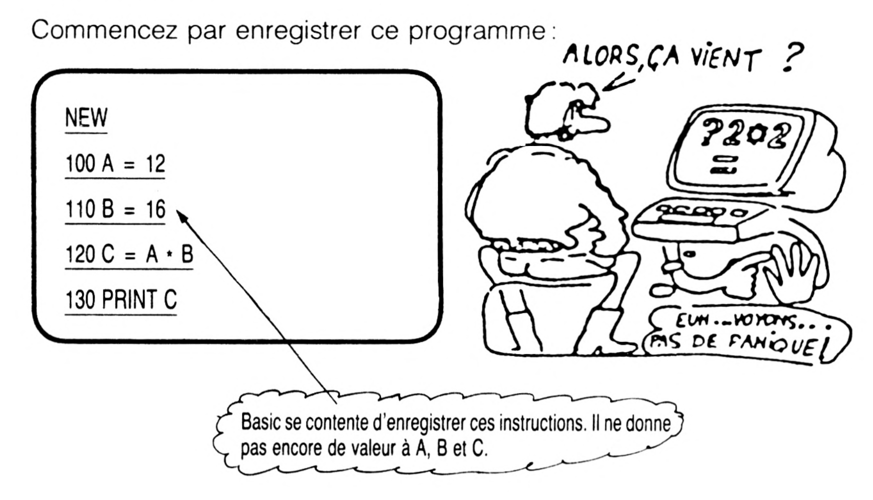


Mais, direz-vous, si je n'aime pas manipuler toutes ces touches et que je préfère retaper toute l'instruction. Cela ne pose aucun problème : il vous suffit de frapper sur « ENTER » sans effectuer aucune correction. Puis vous tapez à nouveau votre instruction.

Un petit programme de multiplication

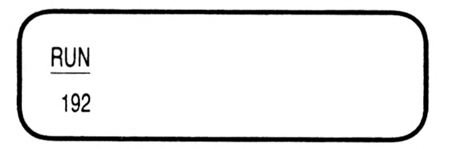
Jusqu'ici, nos petits programmes ne comportaient que des instructions PRINT. Voici maintenant un exemple utilisant des variables et donc des instructions « d'affectation ».

Commencez par enregistrer ce programme :

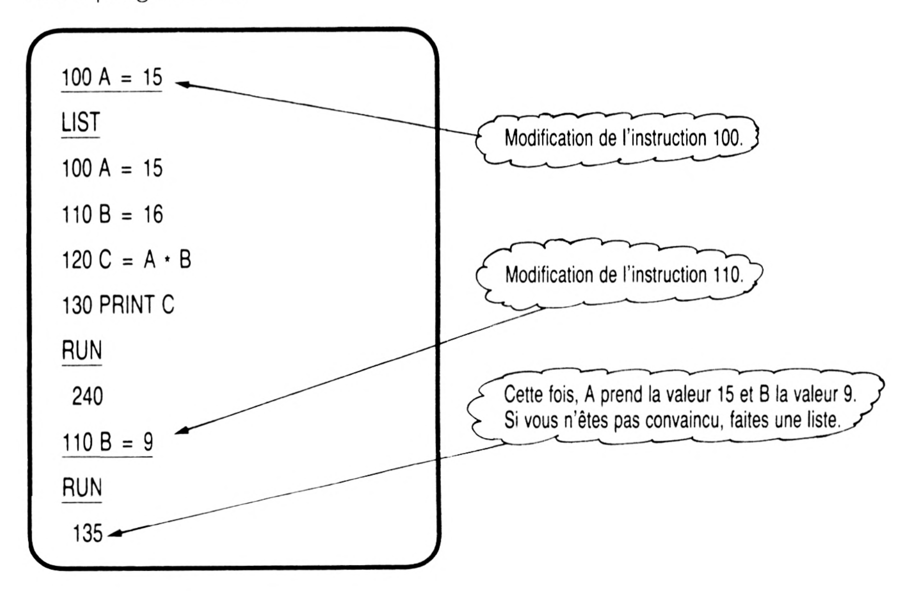


Vérifiez-le en faisant une « liste ». Si vous découvrez une erreur, retapez l'instruction coupable.

Si maintenant nous exécutons ce programme en tapant RUN, Basic commence par exécuter l'instruction 100. Il range donc la valeur 12 dans A. L'instruction 110 lui fait placer 16 dans B. Puis, arrivé en 120, il calcule la valeur de A * B, ce qui fait ici 192, et il la range dans C. Enfin, l'instruction 130 lui fait écrire la valeur de C.



Nous pouvons obtenir d'autres résultats en modifiant une des instructions de notre programme :





Faites-vous la main

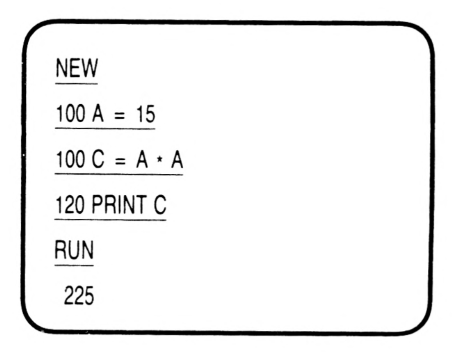
- Tapez ce petit programme (après avoir tapé NEW).

```
100 A = 15
110 B = 12
120 S = A + B
130 D = A - B
140 PRINT S
150 PRINT D
```

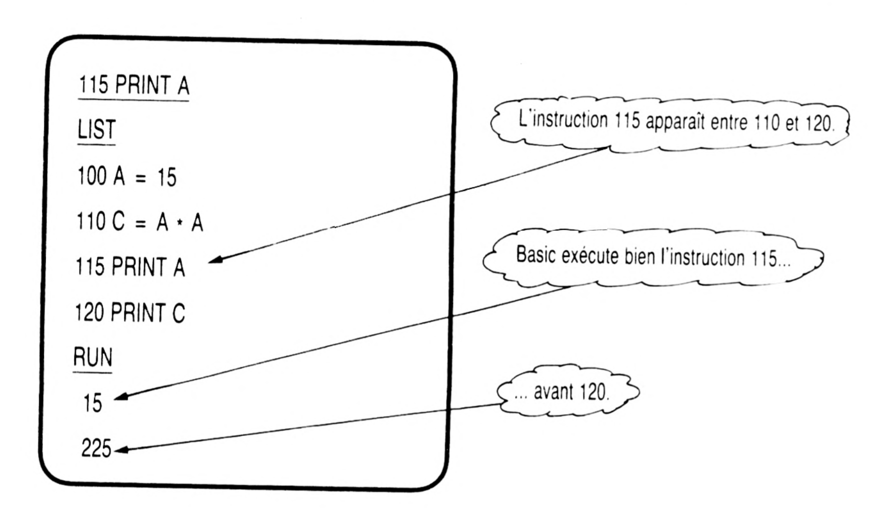
- Faites-en une liste puis exécutez-le.
- Modifiez la ligne 100, en tapant :
  100 A = 255
- Listez et exécutez le nouveau programme ainsi obtenu.
- Même chose avec : 110 B = 148
- Même chose avec : 120 S = A * B

Pour insérer une nouvelle instruction dans un programme

Enregistrez d'abord ce nouveau programme puis exécutez-le.



Enregistrez alors une nouvelle instruction de numéro 115, comme indiqué. Demandez ensuite une liste puis faites exécuter le programme ainsi obtenu :



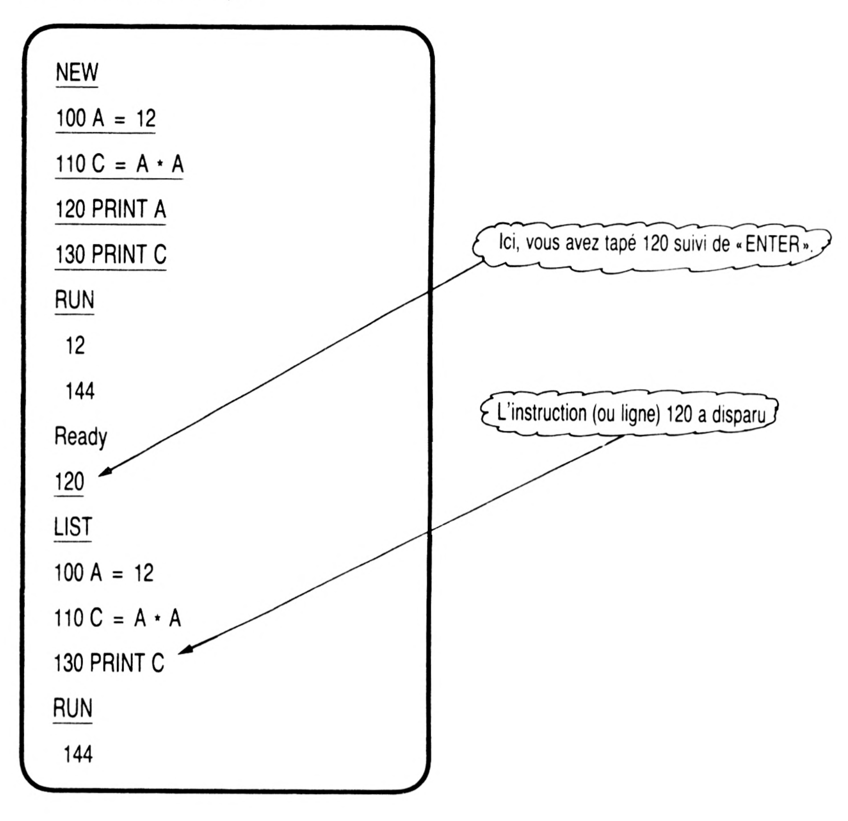
Vous constatez que les instructions apparaissent dans l'ordre de leurs numéros. Basic les exécute dans cet ordre. Ceci vous montre que vous n'êtes pas obligé d'entrer vos instructions dans l'ordre où elles doivent apparaître dans le programme. Pour insérer une nouvelle instruction entre deux autres, il suffit de lui donner un numéro intermédiaire. C'est pour cette raison que nous avons choisi jusqu'ici des numéros allant de 10 en 10. Cela laisse de la place pour insérer de nouvelles lignes en cas de besoin.



Pour supprimer une instruction dans un programme

C'est peut-être la chose la plus simple que l'on puisse faire en Basic. Jugez-en par vous-même : il suffit de taper son numéro, suivi de « RETURN ».

En voici un exemple :





Faites-vous la main

- Tapez NEW et enregistgrez ce programme :
  100 X = 1483
  110 Y = 38
  120 PRINT P
- Listez-le et exécutez-le. Il doit vous imprimer 0. Regardez pourquoi : la variable P ne reçoit pas de valeur.
- Ajoutez l'instruction :
  115 P = X * Y
- Listez et exécutez à nouveau.
- Effacez la ligne 115.
- Tapez ceci :
  120 P = X * Y
  130 PRINT P
- Listez et exécutez.

Et si vous « pataugez » complètement

Si vous êtes débutant, vous devez certainement vous sentir timide et hésitant vis-à-vis du clavier.

Dites-vous bien que quoi que vous fassiez, vous ne risquez pas d'abîmer votre ordinateur. Mais, direz-vous, et si je fais une « fausse manœuvre »... si je bloque tout... si je ne comprends plus où j'en suis... si le mot « Ready » n'apparaît plus...

Dans ce cas, il y a toujours un remède efficace : il suffit d'éteindre votre Amstrad, d'attendre quelques secondes et de le rallumer. Cela vous permet de *« repartir à zéro »*. C'est le *« on efface et on recommence »* bien connu. Bien entendu, vous perdez alors votre programme.

Il existe une autre solution, plus douce. Elle consiste à revenir à l'*« état initial »* en appuyant simultanément sur les trois touches :



Vous constatez que tout se passe comme si vous veniez d'allumer votre Amstrad. L'écran s'efface et vous voyez apparaître le texte de présentation suivi du mot Ready (mais votre programme a quand même disparu).



EXERCICES

(N'oubliez pas de chercher d'abord « sur le papier »)

1. Quelle différence y a-t-il entre ces deux instructions :
   PRINT 25 + 15
   100 PRINT 25 + 15

2. Soit le programme :
   100 N = 15
   110 PRINT N * N
   Comment le modifier pour obtenir :
   a) le carré de 25 ?
   b) le carré de 43 ?

3. Soit le programme :
   100 A = 18
   110 B = 22
   120 PRINT A * A
   a) Comporte-t-il une erreur ? (pour Basic). Que produira son exécution ?
   b) Que contient-il d'anormal ? (pour vous).
   c) Comment lui ajouter une instruction nous fournissant également le carré de B ?

4. Supposez que vous ayez tapé ces instructions :
   NEW
   100 N = 16
   110 PRINT N
   120 PINT N * N
   a) Qu'obtiendrez-vous si vous tapez RUN ?
   b) Comment corriger la ligne 120. Qu'obtiendrez-vous alors ?

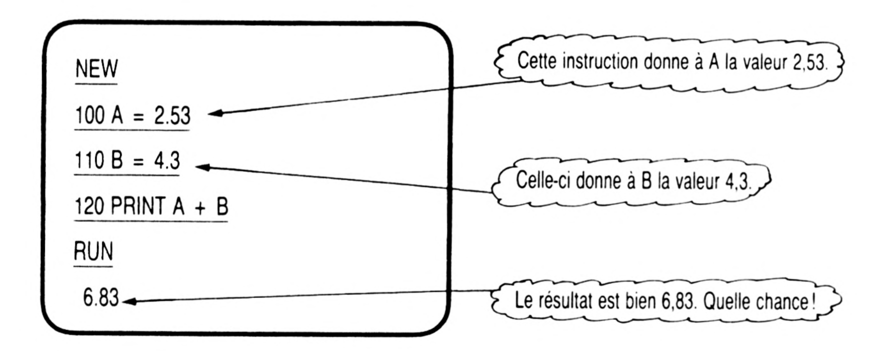
5. Que signifie ce message :
   Syntax error in 140

6. Comment se sortir d'un cas désespéré ?

III. Des chiffres, mais aussi des lettres

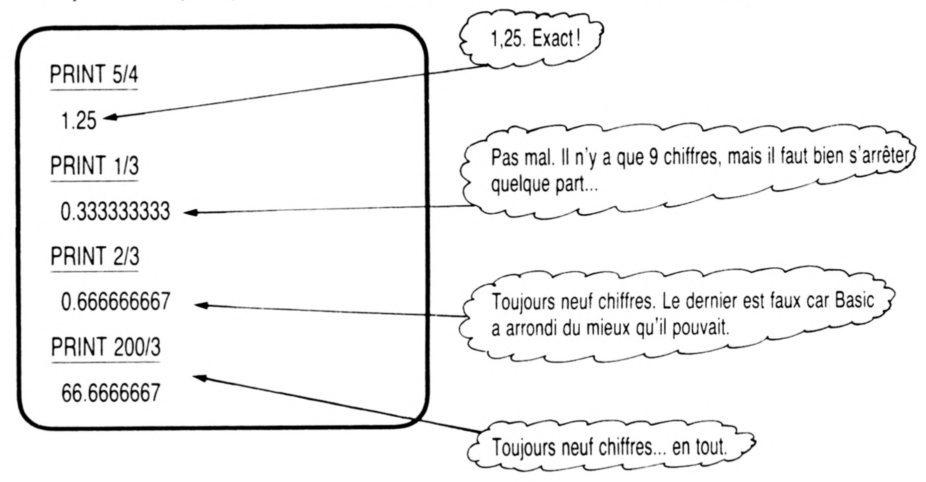
Point ou virgule ?

Jusqu'ici, nous avons travaillé avec des nombres entiers. Mais, bien entendu, Basic peut effectuer des calculs avec des nombres décimaux. Il vous faut simplement penser à remplacer la virgule par un point (comme vous le feriez avec une simple calculatrice). En voici un petit exemple :



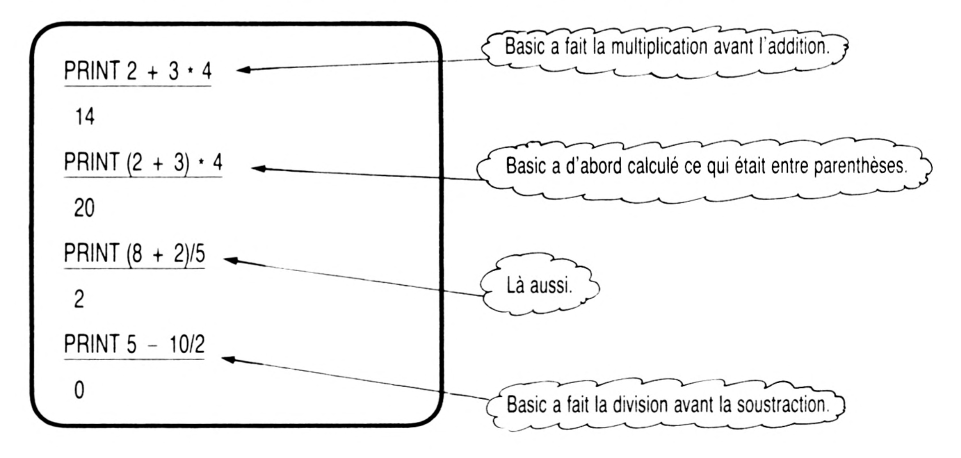
Travail de précision

Essayez ces quelques instructions (en mode «direct» cette fois):



On dit que le Basic de l'Amstrad représente les nombres avec une précision de *«9 chiffres significatifs»*.

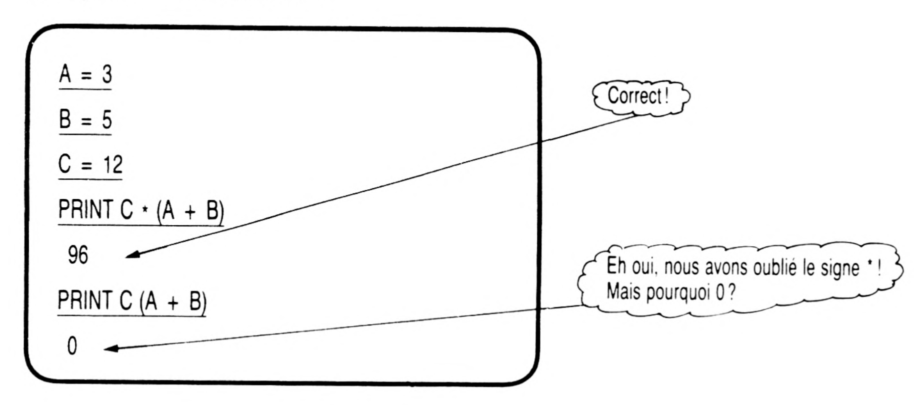
Respectez la priorité ou mettez entre parenthèses:



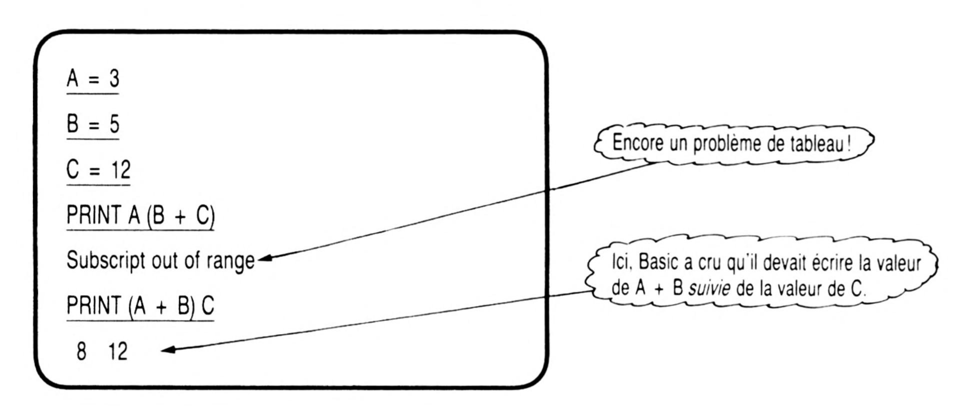
Vous constatez que les règles de priorité sont les mêmes que pour une calculatrice.

Attention aux erreurs

Essayez ces instructions :



Le zéro que nous obtenons semble bien mystérieux. C'est qu'en fait C (A + B) signifie quelque chose pour Basic : C représente alors un « tableau ». Ne cherchez pas à savoir ce qu'est un tableau pour l'instant. Sachez seulement que si vous oubliez un signe * devant ou derrière une parenthèse, vous n'aurez pas toujours droit à un message « Syntax error ». Vous pourrez aussi obtenir d'autres résultats curieux comme dans ces exemples :



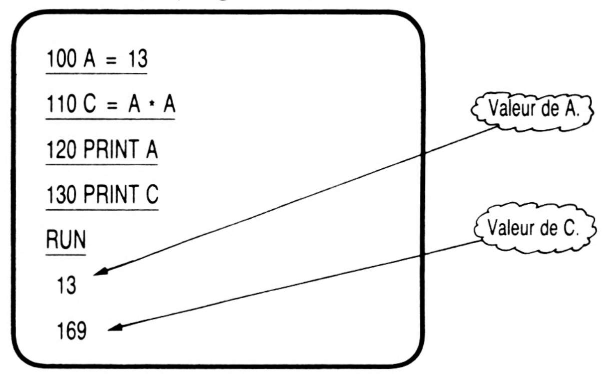
Faites-vous la main

Essayez de prévoir la réponse de Basic dans chacun des cas suivants. Vérifiez ensuite.

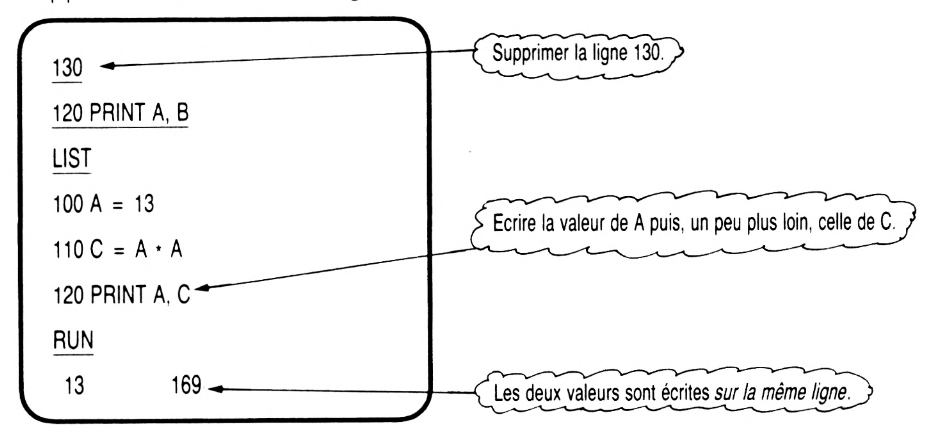
- PRINT 4 * 6 - 5
- PRINT 5 * 3 + 2 * 4
- PRINT (5 + 3) * (8 - 7)
- PRINT (8 + 3) 5
- PRINT 3 (9 + 7)
- A = 12
- B = 20
- PRINT (A + 5) B
- PRINT A (B + 3)
- PRINT 3 * (2 + 5

Pour écrire plusieurs choses sur une même ligne

Examinez ce programme :



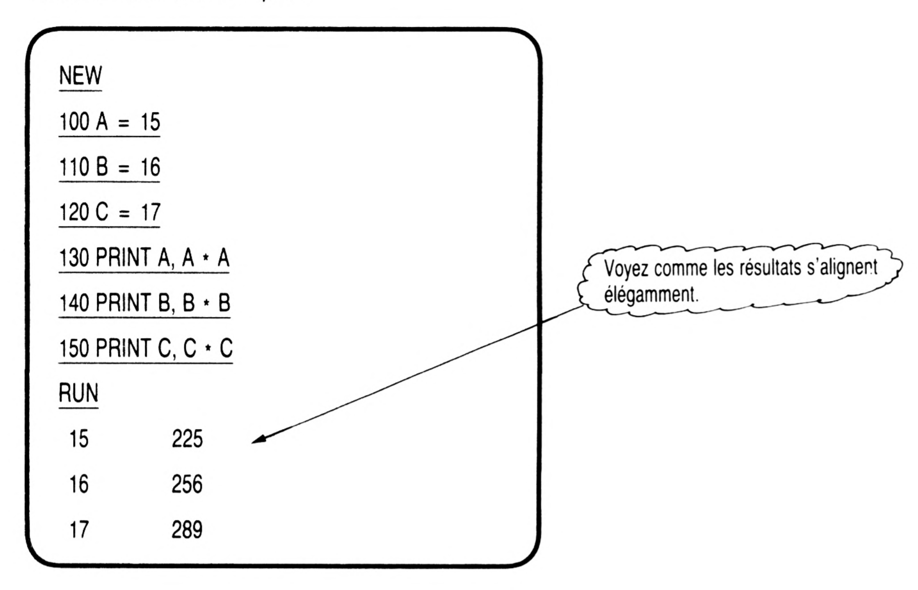
Les valeurs de A et C apparaissent sur *deux lignes différentes*.
Supprimons maintenant la ligne 130 et modifions 120 comme indiqué :



Vous voyez qu'un intervalle assez important sépare les deux valeurs. En fait, Basic partage chaque ligne d'écran en 3 parties égales. La virgule figurant dans l'ordre PRINT demande d'écrire sur la partie suivante de la même ligne.

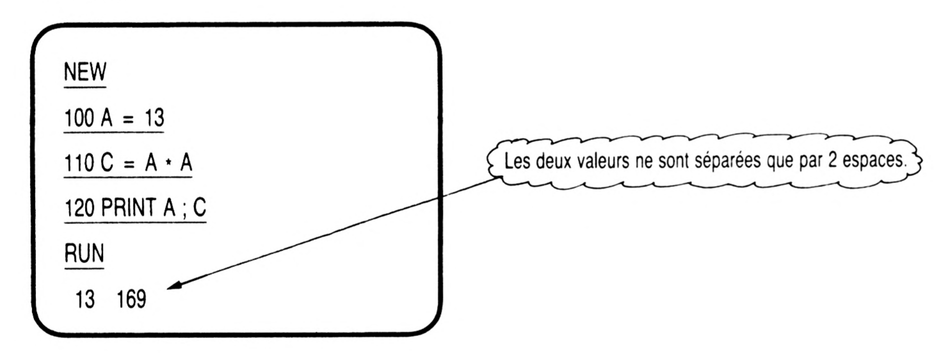
Vous pouvez ainsi faire écrire autant de choses que vous le souhaitez dans une même instruction PRINT. Si, par hasard, vous en écrivez plus que la ligne ne peut en contenir, Basic passera automatiquement à la ligne suivante.

Voici un autre exemple :



Pour tasser...

Vous pouvez aussi utiliser un point-virgule (;) au lieu d'une virgule. Voyez le résultat :



En fait le point-virgule demande d'écrire les choses à la suite l'une de l'autre. Mais, les nombres sont affichés avec un emplacement pour le signe et un espace après. C'est pourquoi il y a quand même deux espaces entre 13 et 169.



Faites-vous la main

* Essayez de prévoir ce que vont produire ces instructions. Vérifiez le

- PRINT 1, 5
- PRINT 5, 10, 15
- PRINT 1.2, 5.43, 4
- PRINT 3 ; 5
- PRINT 2.5 ; 3
- PRINT 2, 5 ; 2, 4

* Mêmes questions avec ce programme

```
100 A = 9
110 B = 5
120 PRINT A, B, A + B
130 PRINT B, 2 * B
140 PRINT A ; B, A * A ; B * B
```

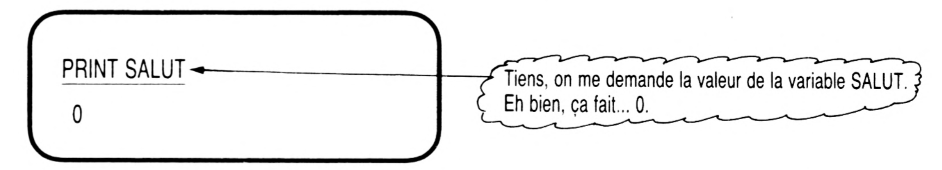
Pour écrire du texte

Vous trouvez maintenant tout à fait normal que l'instruction :

```
PRINT 5
```

écrive la valeur : 5.

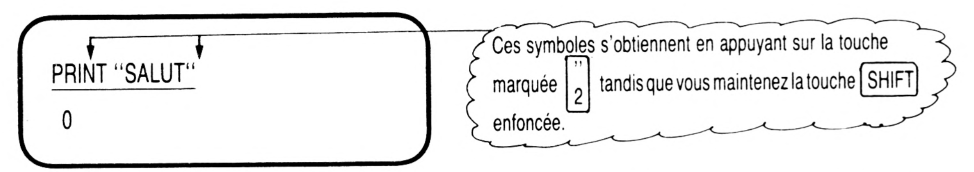
Et si nous voulions écrire autre chose qu'un nombre, par exemple : « SALUT ». Vous songez, un peu hâtivement peut-être à :



En effet, de même que PRINT A signifie « écrire la valeur de A », PRINT SALUT signifie « écrire la valeur de la variable « SALUT ». Comme il est probable que vous ne lui avez encore donné aucune valeur, Basic lui donne un zéro d'office.

Comment alors obtenir ce que nous voulions.

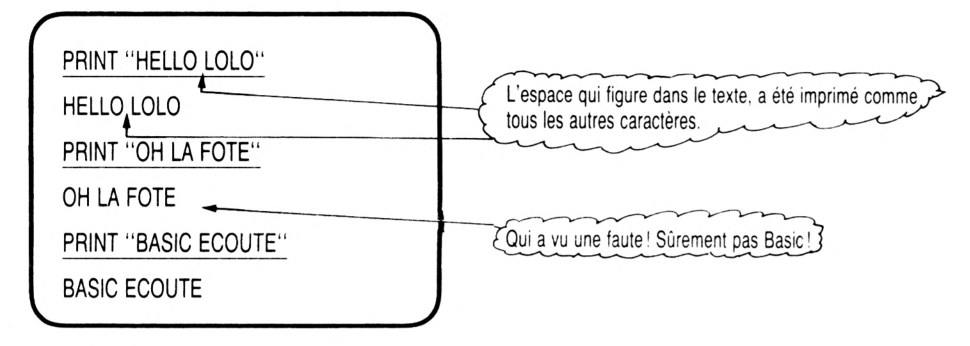
Essayez :



Voilà ! c'est aussi simple que cela. Il vous suffit de placer votre texte entre guillemets.

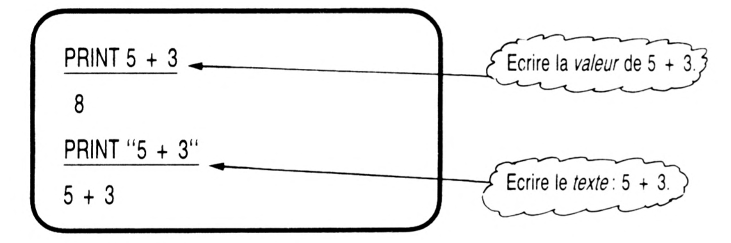
Entre guillemets, pas de censure !

Voici quelques exemples :

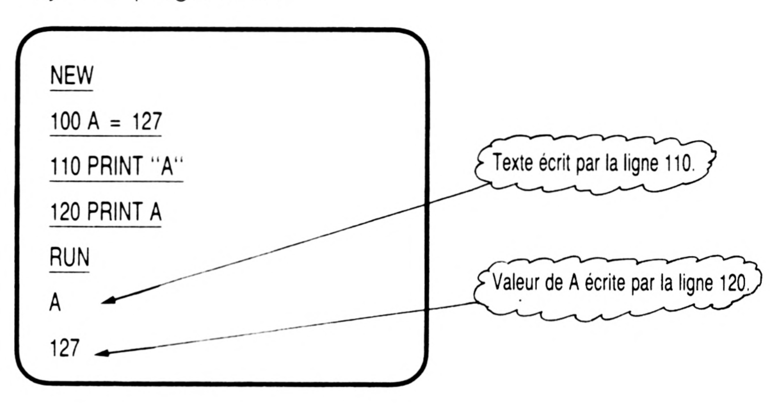


Vous pouvez placer n'importe quel texte entre les guillemets. Basic n'y trouvera jamais rien à dire. En fait, un texte peut contenir n'importe quel caractère que vous trouvez sur le clavier : lettres, chiffres, +, −, =, etc.

Comparez ces deux instructions :



Voyez ce programme :



Faites-vous la main

* Essayez de prévoir ce que vont produire ces instructions. Vérifiez-le.

- PRINT "A = 9"
- PRINT "A = " ; 9
- PRINT "PRINT-EMPS"
- "PRINT 12 + 7"
- PRINT "BONJOUR", "MONSIEUR"
- PRINT "BONJOUR" ; "MONSIEUR"

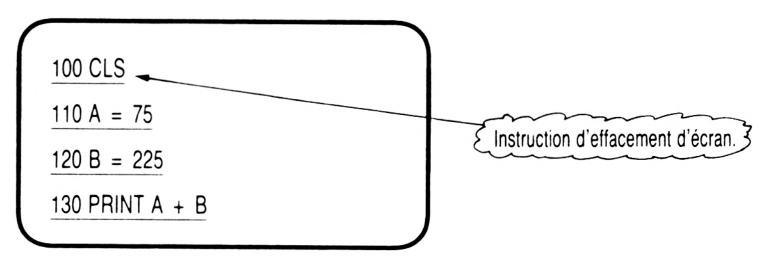
(voyez comme ces deux chaînes sont accolées)

* Mêmes questions avec ce programme :

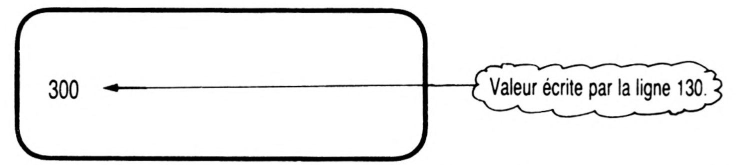
```
100 CLS
110 PRINT "ICI RADIO MIT-MIT"
120 PRINT "LAURENCE ECOUTE"
```

Pour éclairer les résultats d'un programme

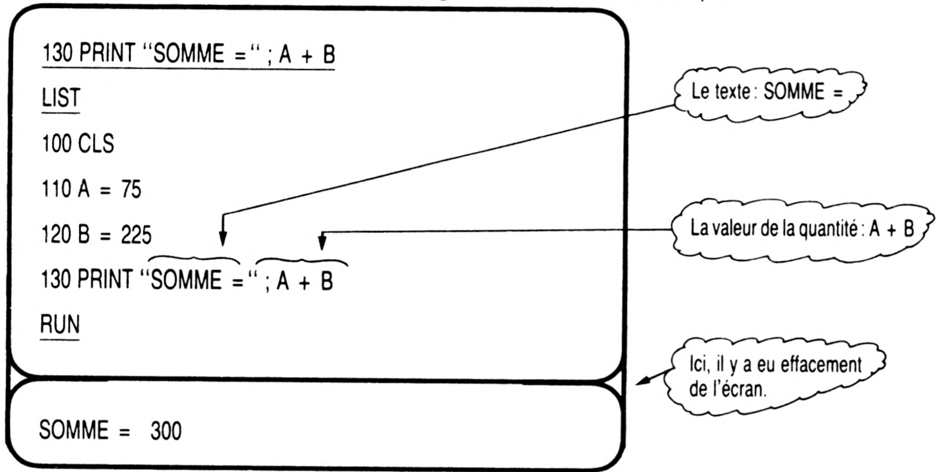
Considérez ce petit programme :



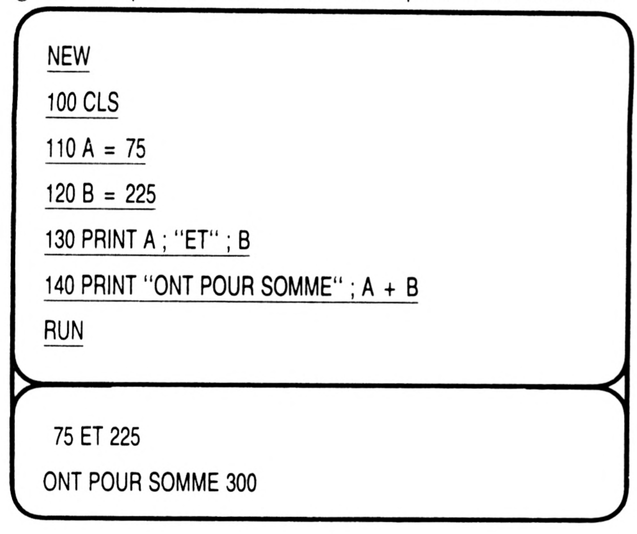
Si vous l'exécutez vous obtenez, en haut de l'écran (que l'instruction 100 a effacé) :



Vous pouvez éclairer ce résultat un peu obscur en l'accompagnant d'un petit texte. Remplacez, par exemple, la ligne 130, comme indiqué :



C'est mieux. Mais on ne sait toujours pas de quelle somme il s'agit. Voici un programme qui fournit des résultats plus satisfaisants :

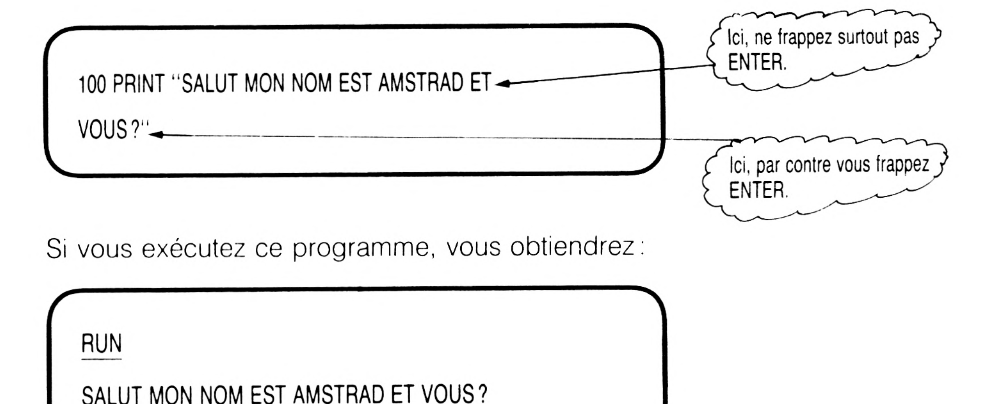


Quand la ligne déborde

Essayez d'enregistrer ce programme d'une seule instruction :

```
NEW
100 PRINT "SALUT MON NOM EST AMSTRAD ET VOUS?"
```

Vous constatez que lorsque vous tapez l'espace entre le ET et le VOUS de la ligne 100, le curseur passe en début de la ligne suivante. En effet, chaque ligne de votre écran ne peut afficher que 40 caractères. Mais rassurez-vous ! Vous pouvez continuer votre instruction. Elle ne se terminera que lorsque vous taperez sur « **ENTER** ». Finalement votre écran se présentera ainsi :



Si vous exécutez ce programme, vous obtiendrez :

```
RUN
SALUT MON NOM EST AMSTRAD ET VOUS?
```

Le texte tient sur une ligne d'écran. Si cela n'avait pas été le cas, Basic l'aurait simplement continué sur la ligne suivante.

Dans la suite de ce livre, nous rencontrerons des instructions ne tenant pas sur une seule ligne d'écran. Nous continuerons à parler de « ligne Basic ». Souvent, nous vous la présenterons « sans coupures », afin qu'elle soit plus lisible.



EXERCICES

1. Quelles sont les deux façons d'écrire sur une même ligne les valeurs de deux variables A et B?

2. Que produira l'exécution de ce programme:
   100 A = 25
   110 B = 5
   120 PRINT A ; "DIVISE PAR" ; B ; "VOT" ; A/B

3. Que fournira l'exécution de ces instructions (en mode direct):
   a) PRINT "5 DIVISE PAR 4 =" ; 5/4
   b) PRINT "5/4 =" ; 5/4
   c) PRINT "2/3 =" ; 5/4

4. Que produira ce programme:
   100 A = 15
   110 B = 8
   120 C = A * B
   130 PRINT "PRODUIT" ; C

   Modifiez-le pour que son exécution fournisse:
   PRODUIT DE 15 PAR 8 = 120

5. Que produira ce programme:
   100 A = 25
   110 PRINT "VOICI UN NOMBRE" ; A
   120 PRINT "VOICI SON DOUBLE" ; 3 * A

IV. Pour établir le dialogue

Pour que votre programme vous interroge

Voici un programme que nous avons déjà rencontré :

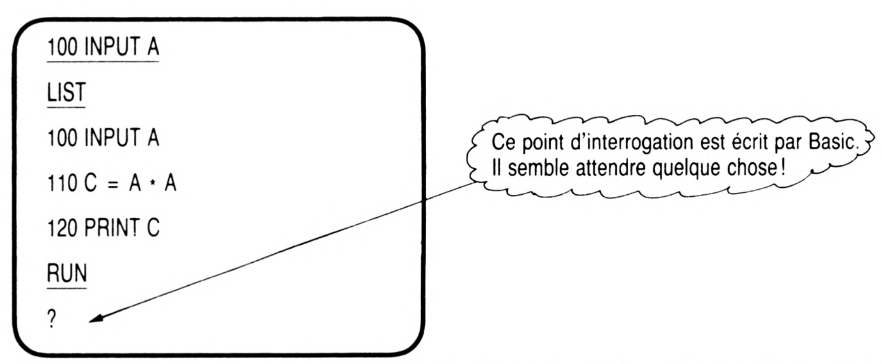
```
100 A = 12
110 C = A * A
120 PRINT C
RUN
 144
```

Il calcule le carré d'un nombre, ici 12. Mais ce nombre est prévu dans le programme. Vous le choisissez lorsque vous *enregistrez* le programme et non pas quand vous *l'exécutez*.

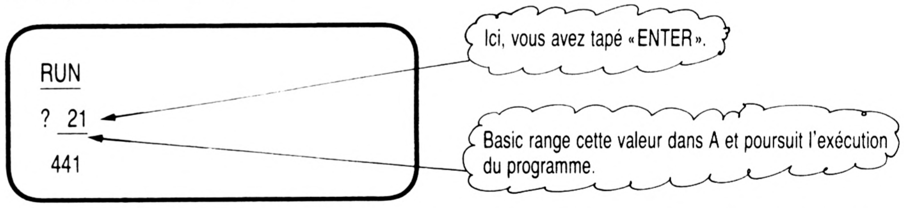
Si vous exécutez plusieurs fois le programme, il vous fournira toujours le même résultat : 144. Et si vous souhaitez obtenir le carré d'autres nombres ? Vous pouvez, bien sûr, modifier l'instruction 100. En voici deux exemples :

```
100 A = 15
RUN
 225
100 A = 25
RUN
 625
```

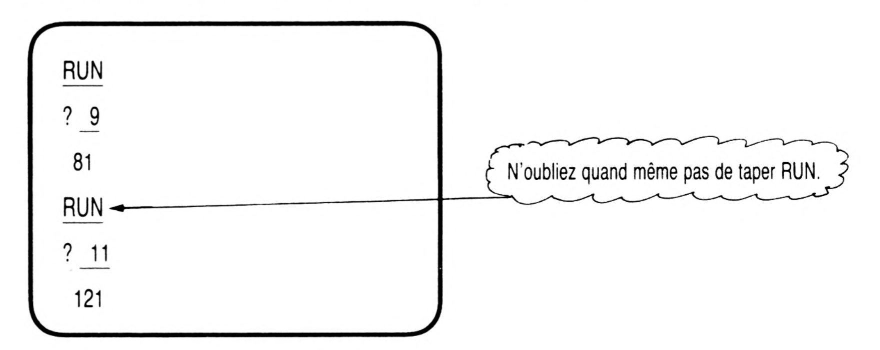
Il existe une instruction qui vous permet de choisir la valeur de A, uniquement au moment de l'exécution du programme. Il s'agit de **« INPUT »** (qui signifie « entrée »). Modifions comme indiqué l'instruction 100 puis exécutons le nouveau programme ainsi obtenu :



A la rencontre de l'instruction INPUT, Basic vous affiche un point d'interrogation pour vous indiquer que vous devez lui fournir une valeur. Il attend patiemment que vous le fassiez. Là encore, vous signalez que vous avez fini de la taper en appuyant sur « ENTER ».



Voyons la souplesse que nous apporte cette instruction en exécutant plusieurs fois notre programme, avec différentes valeurs :

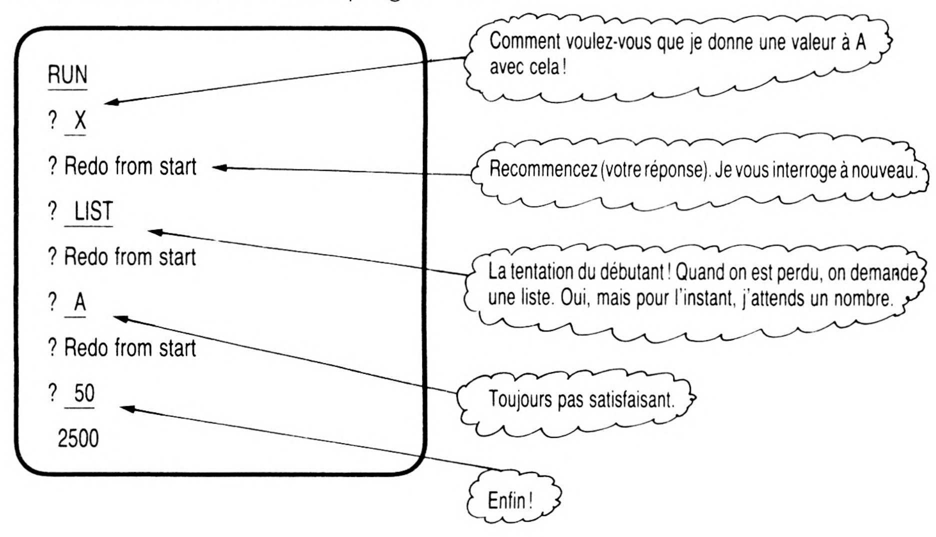


Lorsque vous exécutez ainsi plusieurs fois de suite le même programme, vous risquez d'oublier de frapper RUN. Que se passe-t-il alors si vous tapez directement 11 (alors que Basic ne vous a pas affiché de « ? »).

Tout simplement, Basic effacera la ligne 11. Si elle n'existe pas, ce n'est pas grave, sinon...

En cas de mauvaise réponse

Exécutons à nouveau notre programme :

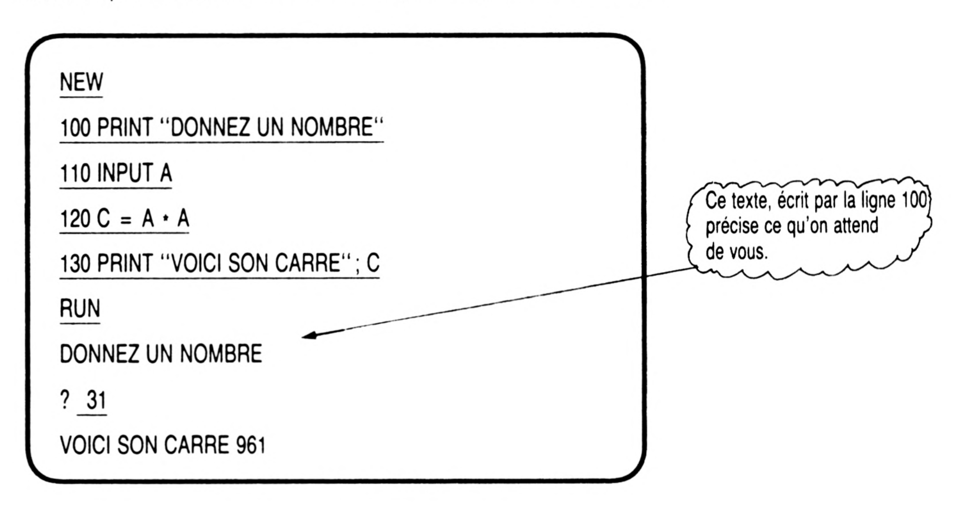


Pour savoir ce que votre programme attend

Lorsque vous voyez apparaître un point d'interrogation, vous ne savez pas toujours ce qu'attend votre programme. Bien sûr, ici, le programme était simple et vous en aviez la liste sous les yeux. Mais imaginez un programme plus compliqué qui demande plusieurs choses ! Supposez qu'en plus, il commence par effacer l'écran !

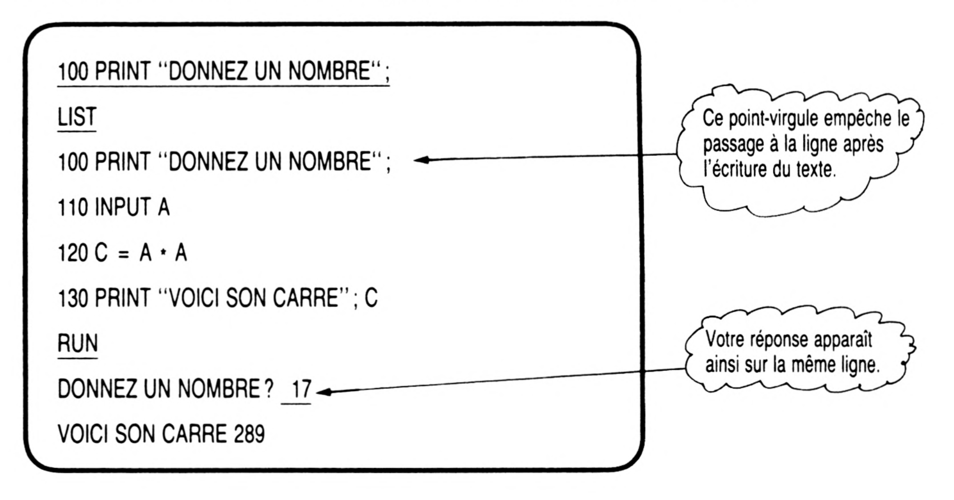


Il est facile de résoudre ce problème. Il vous suffit de faire écrire quelque chose avant que Basic ne vous interroge. Voyez cet exemple :



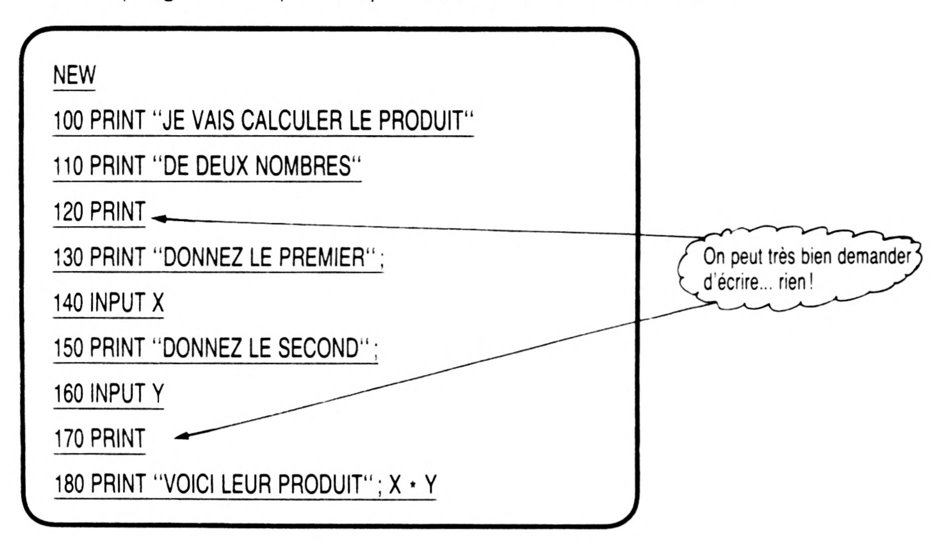
Pour ne pas changer de ligne

Modifions l'instruction 100 en ajoutant un point-virgule à la fin :

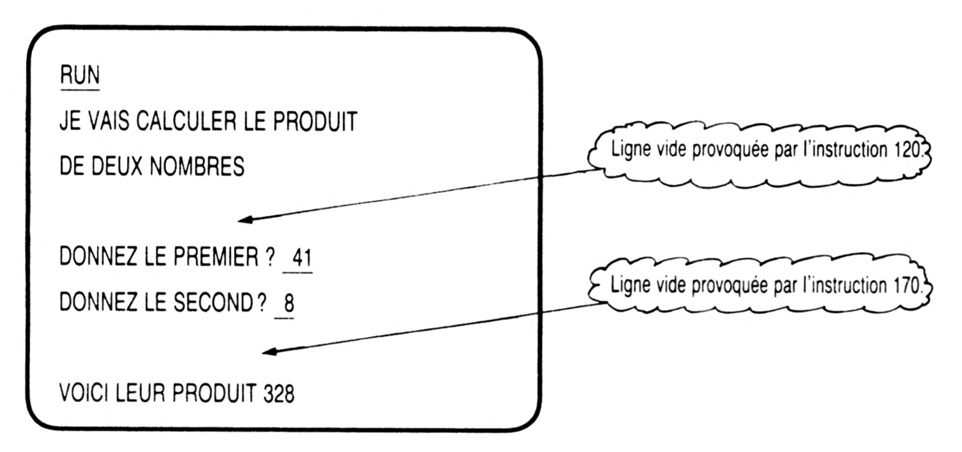


Multiplication à la demande

Voici un programme qui multiplie deux nombres de votre choix :



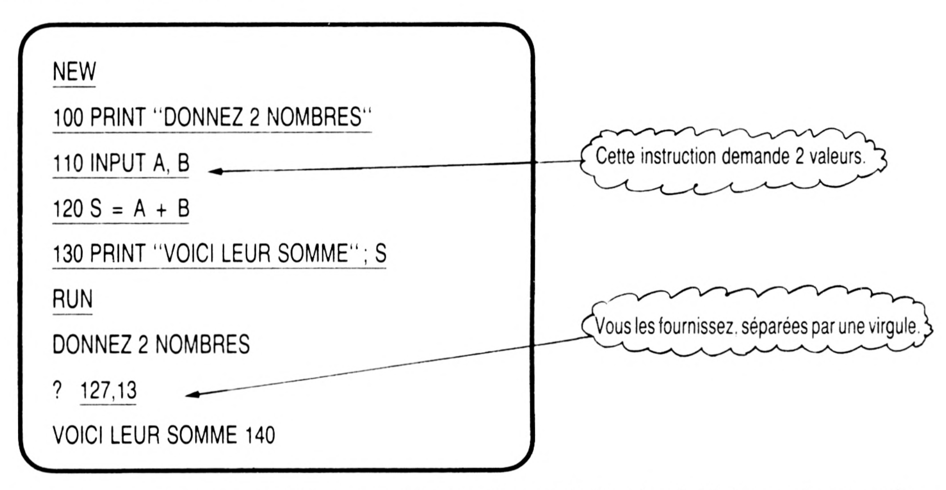
Exécutons-le :



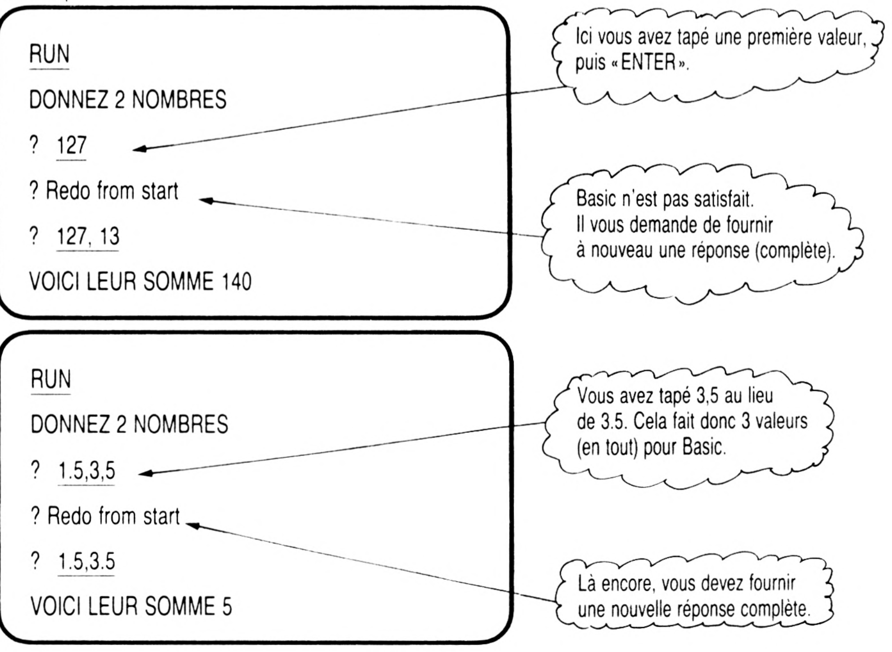
Voyez comme une simple instruction PRINT permet « d'aérer » vos résultats.

Pour fournir plusieurs valeurs dans une même instruction

Il est possible de demander plusieurs valeurs dans une instruction INPUT. Essayez ce programme :



Remarquez que lorsqu'il exécute l'instruction 110, Basic n'affiche toujours qu'un seul « ? ». Il ne vous dit pas combien de valeurs il attend. Vous pouvez facilement vous tromper et en donner trop ou trop peu. Que se passe-t-il alors ? Voici deux exemples :





Faites-vous la main

* Essayez de prévoir ce que fournira ce programme (vérifiez ensuite):
  100 INPUT A, B
  110 PRINT A * B
  lorsque vous répondez:
  - 3, 12
  - 3, 8, 7
  - 4
  - LIST
  - RUN
  - 0,6

Pour manipuler des textes: les variables chaîne

Voyez ce programme:

```
100 A = 8
110 PRINT A
RUN
 8
```

Ranger la valeur 8 dans une variable nommée A. (100 A = 8)

Ecrire la valeur de A. (110 PRINT A)

et celui-ci:

```
100 T$ = "HELLO"
110 PRINT T$
RUN
HELLO
```

$ s'obtient avec la touche [$/4] et [SHIFT]

Ranger le texte «HELLO» dans une variable nommée T$. (100 T$ = "HELLO")

Ecrire le contenu de T$. (110 PRINT T$)

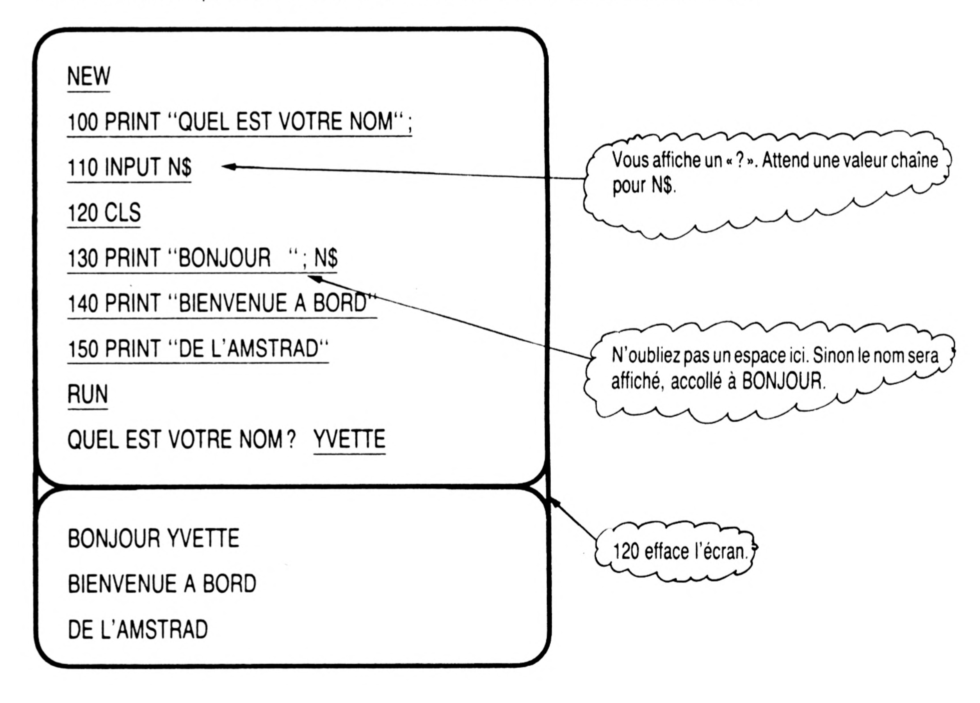
HELLO
T$

Jusqu'ici, nous avons dit que HELLO était un texte. En Basic, le terme exact est chaîne de caractères, ou plus brièvement « *chaîne* ». Nous dirons que T$ est une « *variable chaîne* ».

Comment Basic reconnaît-il une variable chaîne? Simplement au fait que son nom se termine par $. Les règles de formation des noms des variables chaînes sont les mêmes que pour les variables « numériques » (celles qui contiennent des nombres).

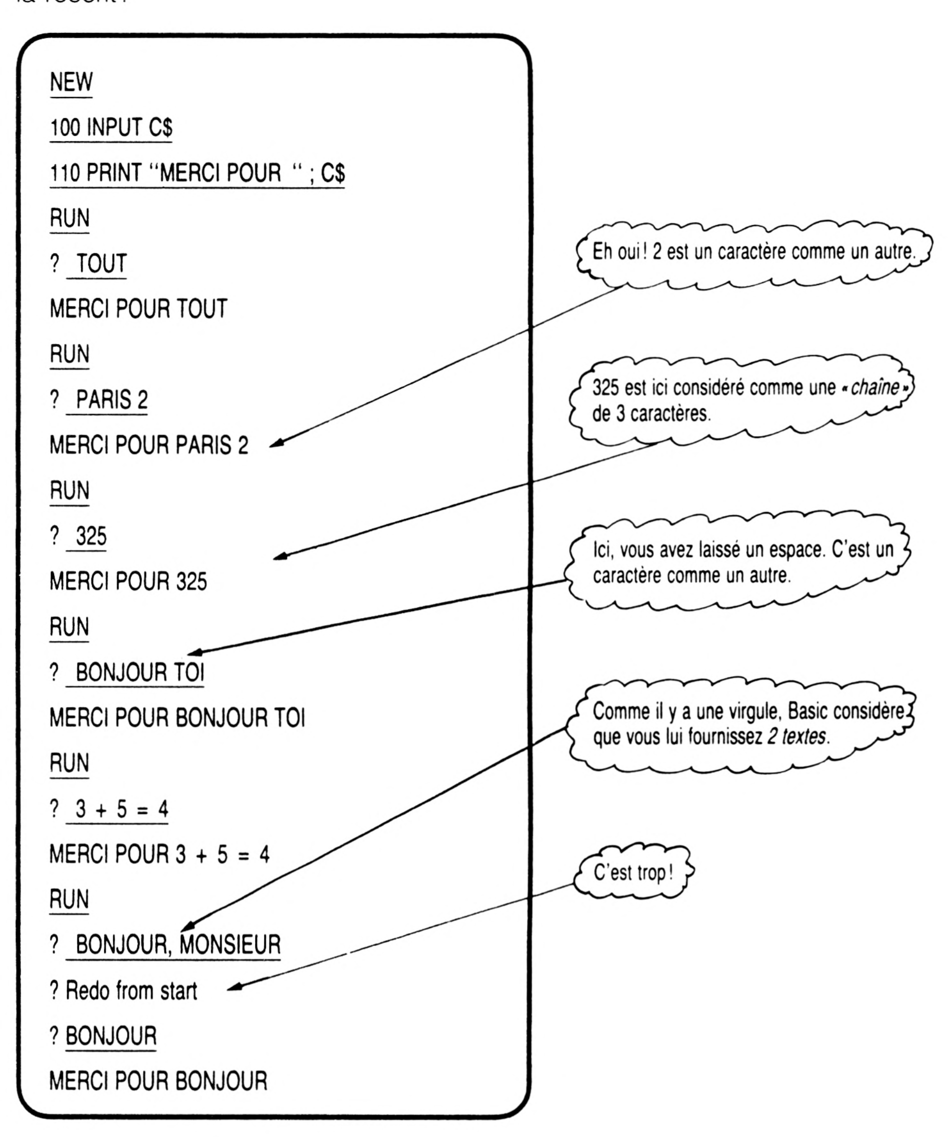
Un programme d'accueil

Voici un exemple amusant d'utilisation d'une variable chaîne :



Les risques d'erreur

Essayez ce programme qui vous demande une chaîne de caractères et qui vous la réécrit :



Enregistrez ce programme.

```
NEW
100 PRINT "QUEL EST VOTRE PRENOM";
110 INPUT PR$
120 PRINT "EN QUELLE ANNEE SOMMES-NOUS";
130 INPUT AN
140 PRINT
150 PRINT "BONJOUR"; PR$
160 PRINT "EN QUELLE ANNEES ETES-VOUS NE";
170 INPUT NA
180 AG = AN - NA
190 CLS
200 PRINT "DITES-MOI "; PR$
210 PRINT "QUE RESSENT-ON"
220 PRINT "QUAND ON A"; AG; "ANS"
```

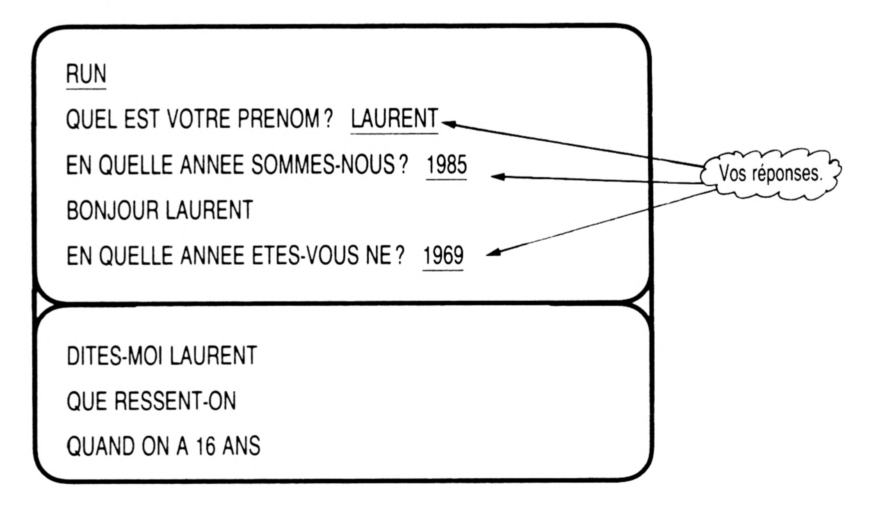
PR$: votre prénom.

AN: année en cours.

NA: votre année de naissance

Calcule votre âge dans AG.

Exécutez-le :



Faites-vous la main

* Enregistrez ce programme :

```
100 INPUT CH$
110 PRINT "VOUS AVEZ DIT"; CH$
```

Voyez ce que produit son exécution lorsque vous répondez :

- BONJOUR
- 25
- BONJOUR, MONSIEUR
- BONJOUR MONSIEUR

* Même questions avec ce programme :

```
100 INPUT N, TX$
110 PRINT "NOMBRE"; N
120 PRINT "TEXTE"; TX$
```

et les réponses suivantes :

- 127, BONJOUR
- 485, 29
- N, 32
- 47
- 212, BONJOUR, 35
- 212, BONJOUR, MONSIEUR



EXERCICES

1. Vrai ou faux ? Un programme fournit toujours les mêmes résultats.
2. Que signifie l'apparition d'un point d'interrogation ?
3. Que produira ce programme :

```
100 INPUT A
110 PRINT "MERCI POUR"; A
```

suivant que vous l'exécutez en répondant :
a) 25
b) BONJOUR
c) LIST

4. Mêmes questions avec :

```
100 INPUT A$
110 PRINT "MERCI POUR   "; A$
```

5. Que produit ce programme :

```
100 PRINT "BONJOUR";
110 PRINT "MONSIEUR"
```

6. Que produira ce programme :

```
100 PRINT "BONJOUR"
110 PRINT
120 PRINT
130 PRINT "MONSIEUR"
```

7. Que signifie ce message :

```
? Redo from start
```

V. Bouclez... bouclez

La boucle infernale

Enregistrez ce petit programme :

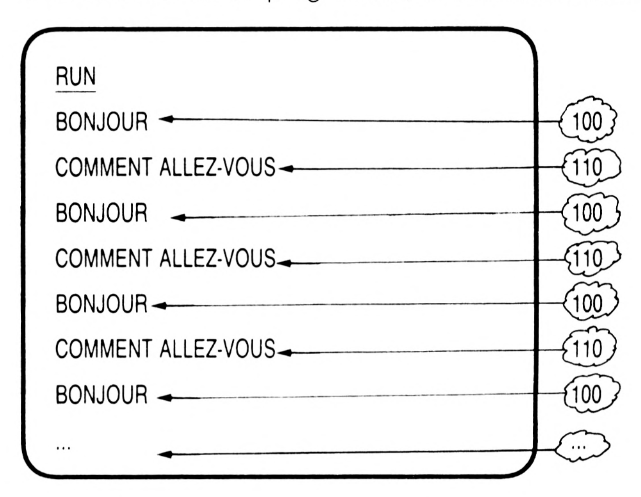
```
NEW
100 PRINT "BONJOUR"
110 PRINT "COMMENT ALLEZ-VOUS"
RUN
BONJOUR
COMMENT ALLEZ-VOUS
```

Bien sûr, les instructions 100 et 110 ne s'exécutent qu'une seule fois. Ajoutez une nouvelle instruction :

```
120 GOTO 100
LIST
100 PRINT "BONJOUR"
110 PRINT "COMMENT ALLEZ-VOUS"
120 GOTO 100
```

Signifie : aller en 100 (ou revenir en 100 - comme vous préférez).

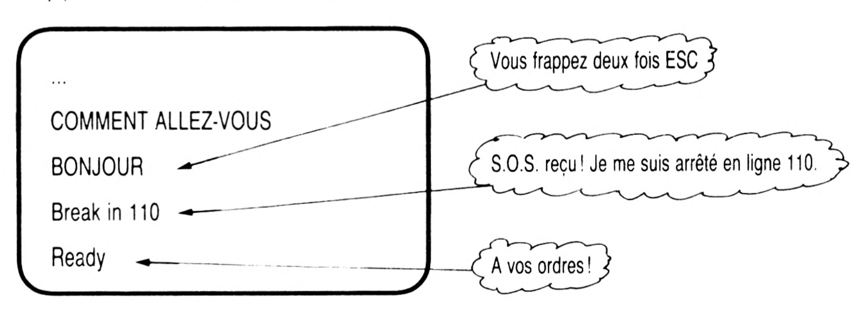
Si vous exécutez ce programme, il va écrire sans cesse le même texte :



Un peu monotone ! Et, en plus, direz-vous, comment s'arrête-t-on ? Appuyez sur la touche :



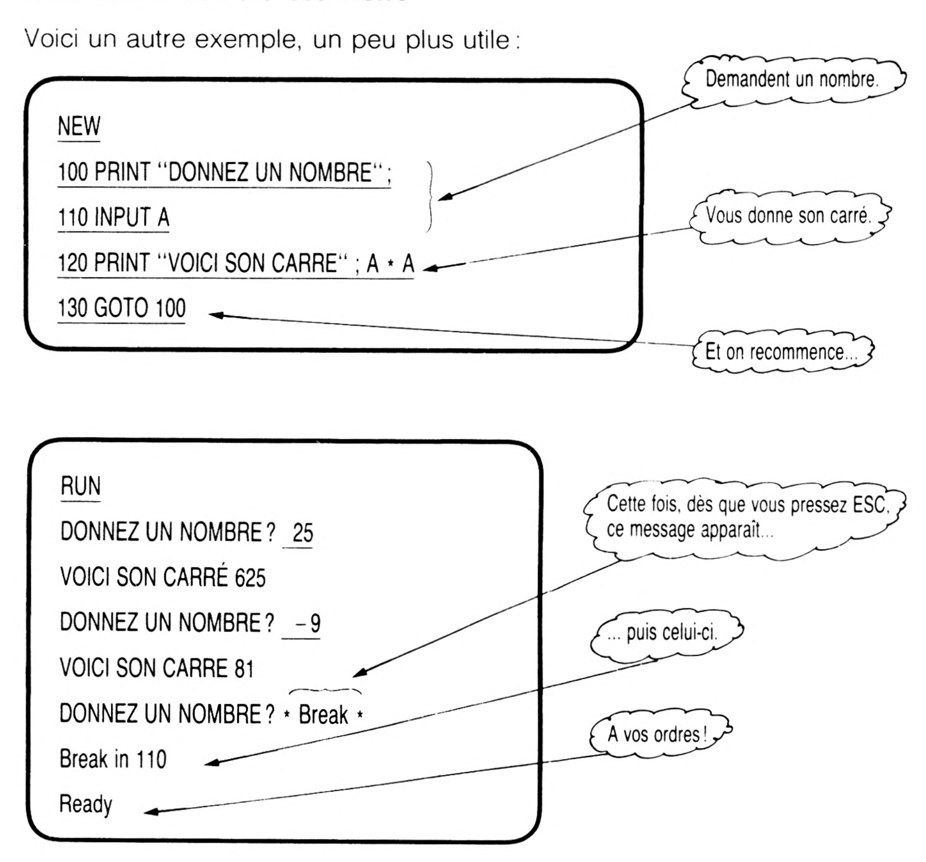
Vous voyez que l'exécution s'interrompt mais Ready n'apparaît pas. Si vous tapez sur n'importe quelle touche, l'exécution reprend. Par contre, essayez de frapper 2 fois de suite sur ESC.



Basic vous indique quelle ligne il exécutait lorsque vous l'avez interrompu. Dans votre cas, ce sera peut-être 100...

Une autre boucle éternelle

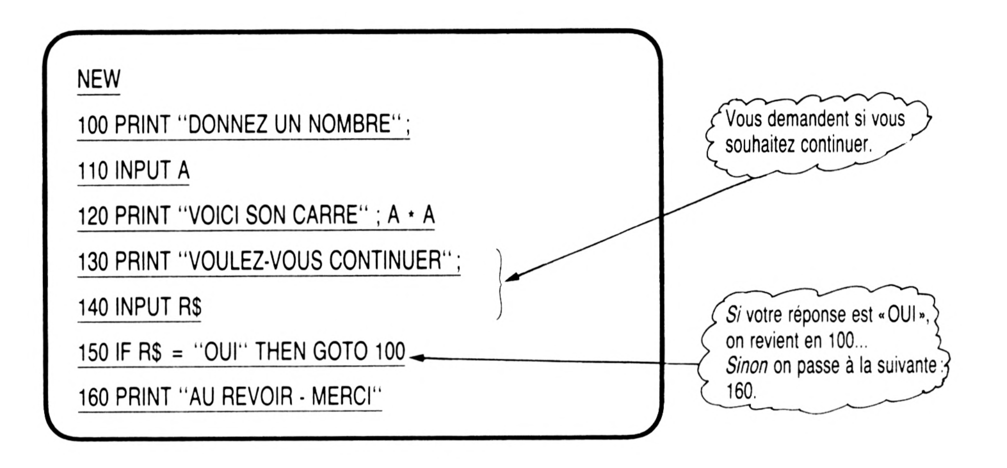
Voici un autre exemple, un peu plus utile :



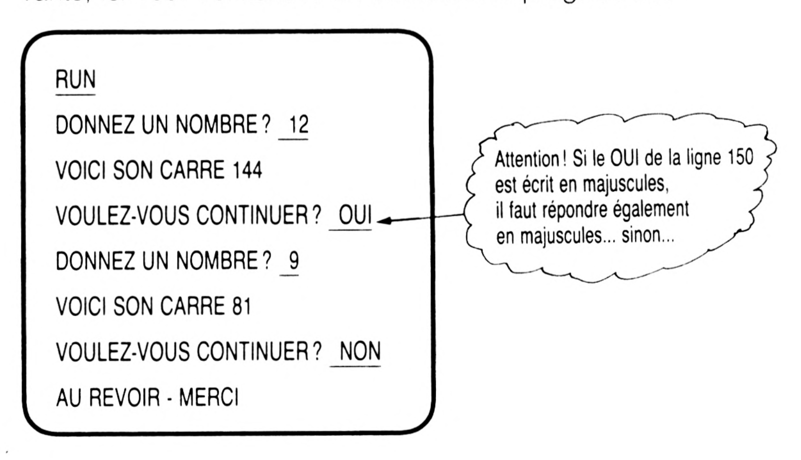
Pour contrôler la boucle

L'instruction GOTO 100 renvoie systématiquement Basic à la ligne 100. On dit que c'est un « *branchement inconditionnel* ». Inconditionnel, cela veut dire « sans aucune condition ».

Mais il existe une instruction de « branchement conditionnel » ; dans ce cas le « branchement » n'a lieu que *si* une certaine condition est réalisée. Nous employons une telle instruction dans l'exemple suivant :

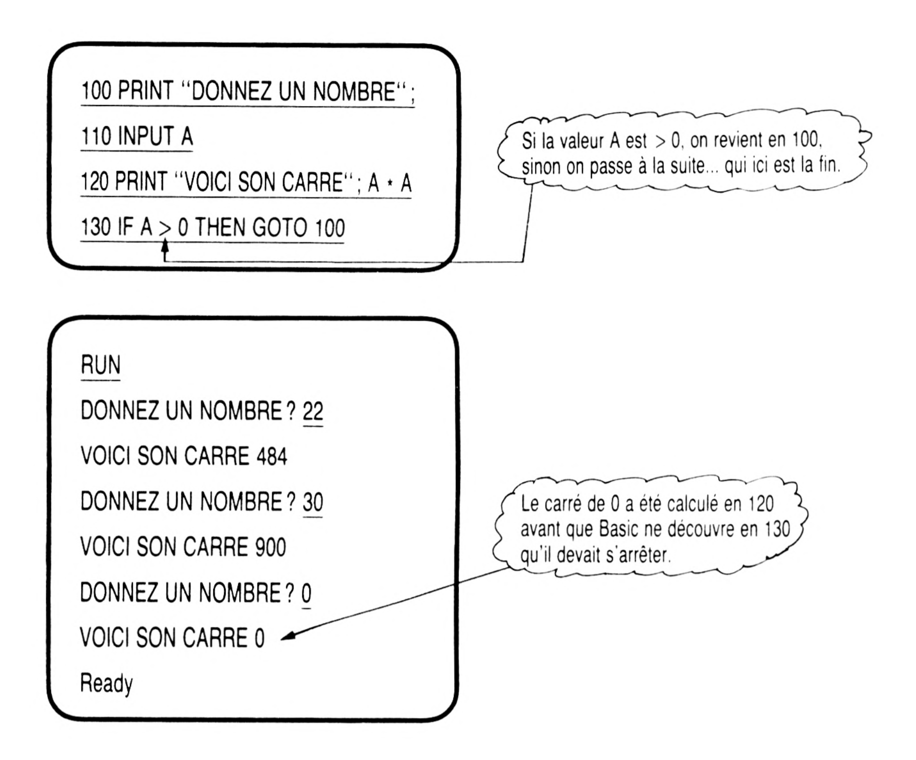


L'instruction 150 signifie exactement : *Si* (IF) la variable R$ contient la valeur «OUI», alors revenir en 100 ; dans le cas contraire, passez à l'instruction suivante, ici 160. Vérifions-le en exécutant le programme :



Une autre façon de s'arrêter

Au lieu de poser la question « VOULEZ-VOUS CONTINUER », nous pouvons convenir de ne continuer que lorsque la valeur fournie pour A est positive :



Comparez des choses comparables

Dans nos deux exemples, nous avons rencontré deux sortes de conditions :

- R$ = « OUI » La variable chaîne R$ contient-elle le texte « OUI » ?
- A > 0 La variable numérique A contient-elle une valeur supérieure à zéro ?

Nous aurons l'occasion de rencontrer d'autres sortes de comparaisons par la suite. Pour l'instant, sachez que Basic refusera de comparer des chaînes avec des nombres. Voyez ce qui se passerait si vous cherchiez quand même à le faire :

```
NEW
100 INPUT R$
110 INPUT N
120 IF R$ = N THEN GOTO 100
RUN
? BONJOUR
? 55
Type mismatch in 120
Ready
RUN
? 55
? 40
? Type mismatch in 120
```

Un texte pour R$. (→ BONJOUR)

Un nombre pour N. (→ 55)

Signifie : variables de nature (type) différente en 120. C'est vrai : R$ est une variable *chaîne* ; N est une variable *numérique*. (→ Type mismatch in 120)

Ici, ce sont les *2 caractères* « 55 » qui sont placés dans R$. (→ 55)

La comparaison est toujours impossible. (→ Type mismatch in 120)



Faites-vous la main

Essayez de prévoir ce que font chacun des programmes suivants. Vérifiez ensuite en les exécutant sur votre Amstrad.

- ```
  100 PRINT "DONNEZ UN PETIT NOMBRE";
  110 INPUT N
  120 IF N > 30 THEN GOTO 100
  130 PRINT "VOILA"
  ```
- ```
  100 PRINT "DITES MERCI"
  110 INPUT R$
  120 IF R$ <> "MERCI" THEN GOTO 100
  130 PRINT "DE RIEN"
  ```

  <> (caractère < suivi de >) signifie *différent de.*
- ```
  100 PRINT "DITES MERCI"
  110 INPUT R$
  120 IF R$ = "MERCI" THEN GOTO 100
  130 PRINT "JE VOUS AI BIEN EU"
  ```
  ```

Pour compter les tours

Reprenons notre programme de la page 59.

```
NEW
100 PRINT "DONNEZ UN NOMBRE";
110 INPUT A
120 PRINT "VOICI SON CARRE'; A * A
130 IF A > 0 THEN GOTO 100
```

Il nous calcule des carrés jusqu'à ce que nous lui fournissions une valeur nulle (ou négative). Nous allons demander au programme de nous dire combien de carrés il a calculé. Pour cela, nous pouvons « compter » le nombre de fois que Basic répète les instructions 100 à 120.

Comment compter ? En utilisant une variable et en s'arrangeant pour que sa valeur augmente de 1 à chaque tour.

Quelle est l'instruction qui fait augmenter de 1 la valeur d'une variable, par exemple I ? Rappelez-vous le rôle d'une instruction d'affectation comme :

C = A + B

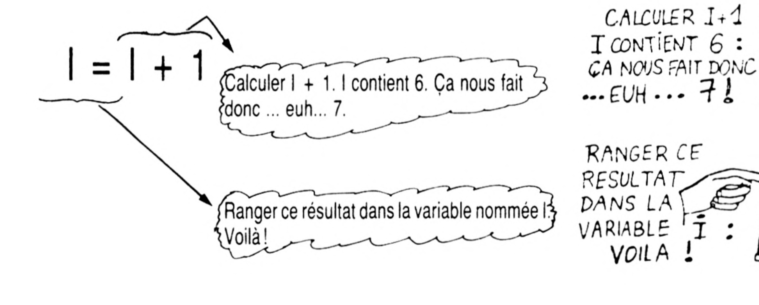
Elle signifie : • calculer la valeur de A + B
• *puis* ranger le résultat dans C

Mais alors que fait :

I = I + 1

Cette instruction signifie : • calculer la valeur de I + 1
• *puis* ranger le résultat dans I

Supposons que I contienne la valeur 6. Voici un schéma qui montre ce que va faire Basic lorsqu'il rencontera l'instruction : I = I + 1



Cette instruction résout bien notre problème de comptage. Incorporons-là dans notre programme, par exemple entre 120 et 130 :

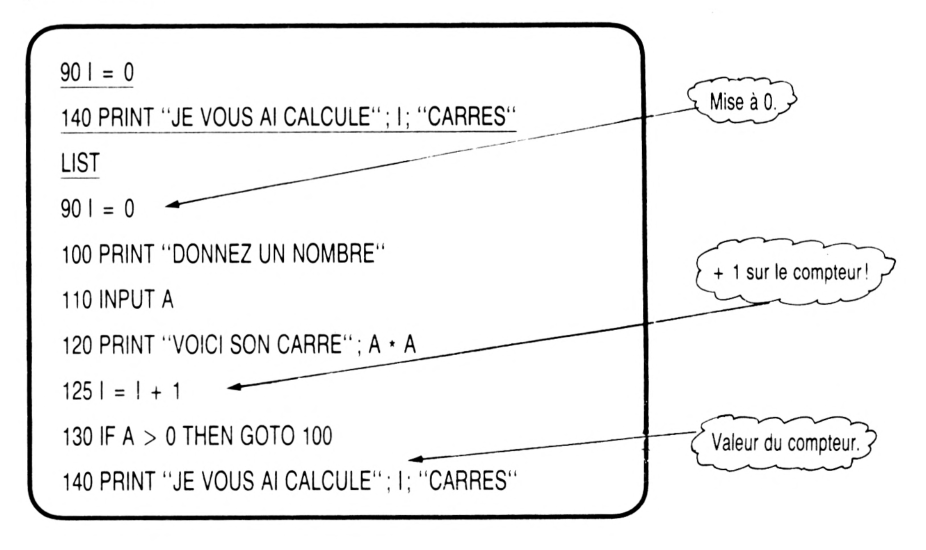
```
125 I = I + 1
LIST
100 PRINT "DONNEZ UN NOMBRE";
110 INPUT A
120 PRINT "VOICI SON CARRE"; A * A
125 I = I + 1
130 IF A > 0 THEN GOTO 100
```

Mais direz-vous, quelle valeur donner au compteur avant de commencer ? Il suffit de le « mettre à zéro » avec une instruction telle que :

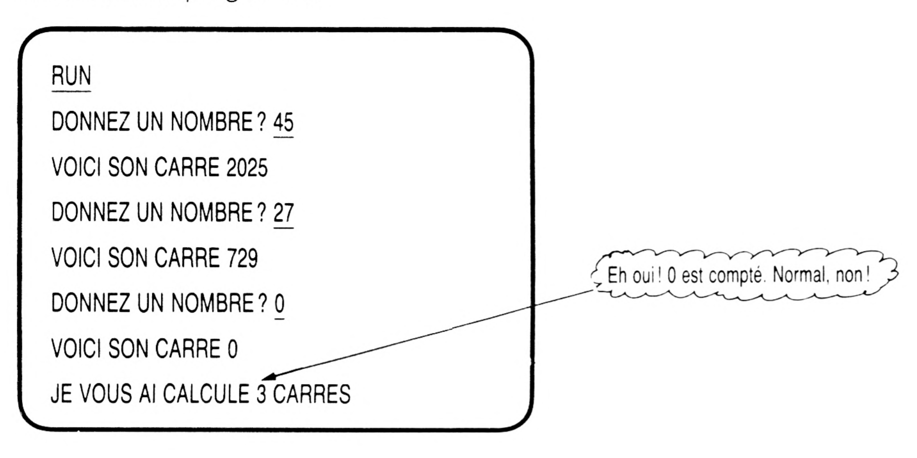
```
I = 0
```

Il faut placer cette instruction avant la ligne 100. Nous lui choisissons par exemple le numéro 90.

Enfin, nous voulons faire écrire la valeur de I lorsque le programme « sort » de la boucle. Pour cela, nous ajoutons une instruction après la ligne 130, par exemple en 140 :

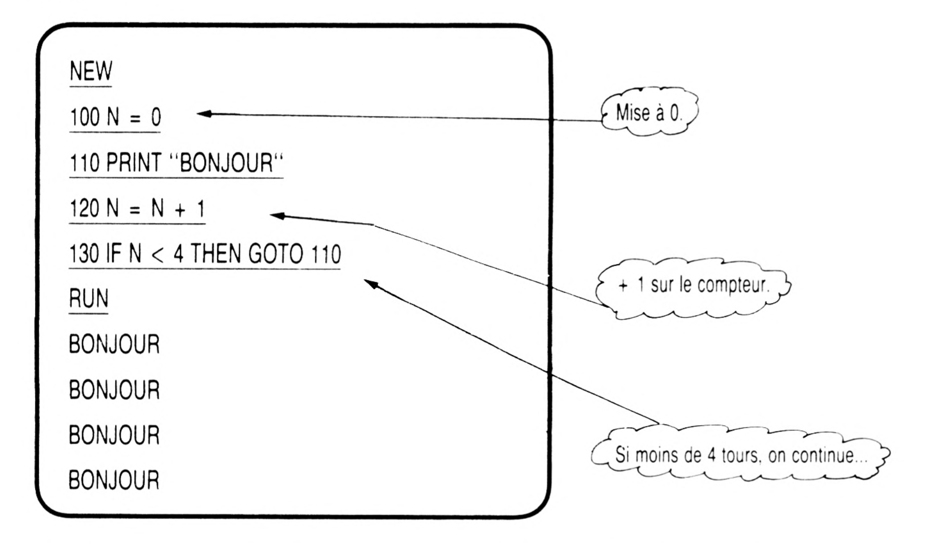


Exécutons ce programme :

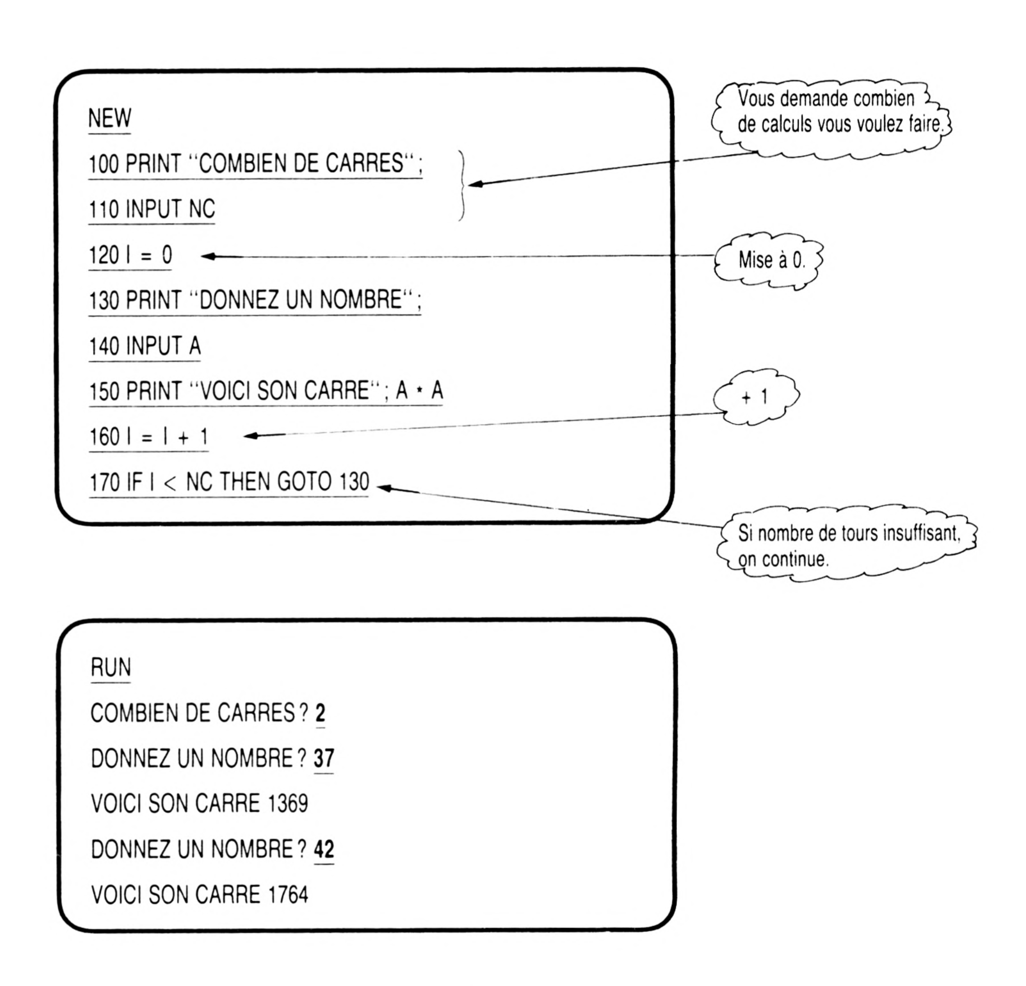


Pour faire le nombre de tours que l'on veut

Maintenant que nous savons compter, nous pouvons également imposer un nombre de tours de boucle. Voyez ce programme qui écrit quatre fois BONJOUR :



Vous pouvez aussi écrire votre programme de façon à ce que Basic vous demande le nombre de tours que vous voulez faire. Ainsi ce programme vous calcule autant de carrés que vous le souhaitez :



Quand on ne sait pas compter

Utiliser un compteur, c'est bien ! Encore faut-il bien placer :

— l'instruction qui lui donne sa valeur « initiale » (0 dans nos précédents exemples).

— l'instruction qui fait avancer le compteur.

Faites-vous la main

* Essayer de prévoir ce que font chacun de ces programmes. Vérifiez-le.

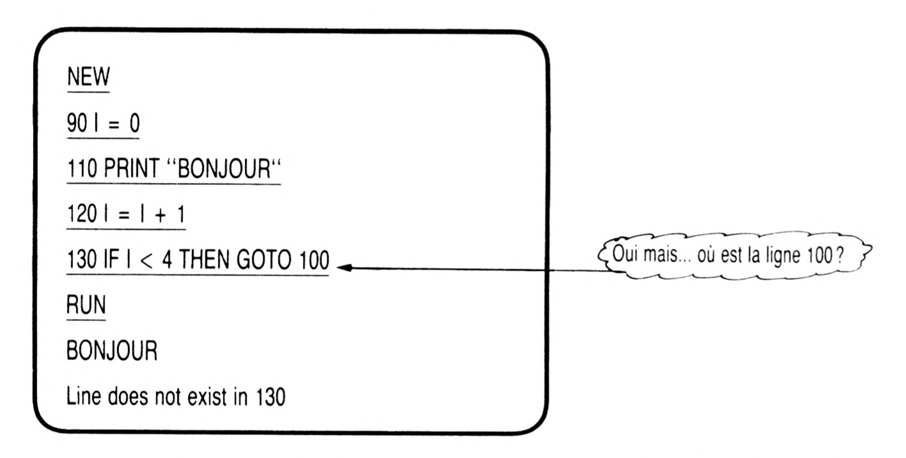
- NEW
```
100 N = 0
110 PRINT "BONJOUR"
120 N = N + 1
130 IF N < 4 THEN GOTO 100
```

- NEW
```
100 N = 0
110 PRINT "BONJOUR"
120 N = N + 2
130 IF N < 4 THEN GOTO 110
```

- NEW
```
100 N = 0
110 PRINT "BONJOUR"»
120 IF N < 4 THEN GOTO 100
130 N = N + 1
```

Restez branchés

Comment va réagir Basic si vous lui demandez de se brancher à une ligne qui n'existe pas? Essayez ce programme :



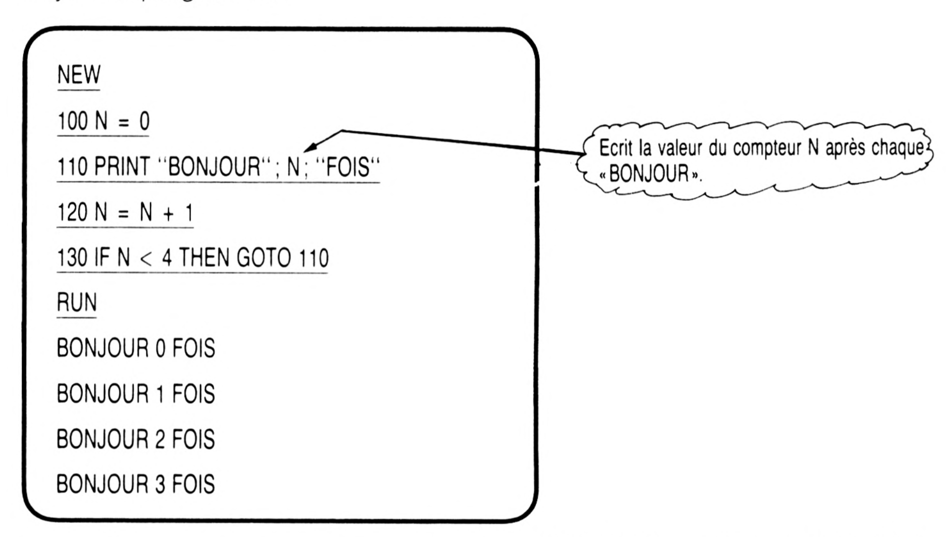
```
NEW
90 I = 0
110 PRINT "BONJOUR"
120 I = I + 1
130 IF I < 4 THEN GOTO 100
RUN
BONJOUR
Line does not exist in 130
```


Basic exécute d'abord les lignes 90, 110 et 120. L'instruction 130 lui demande de revenir en 100. Comme cette ligne n'existe pas, Basic vous le précise par un message qui signifie :

numéro de ligne inexistant en 130

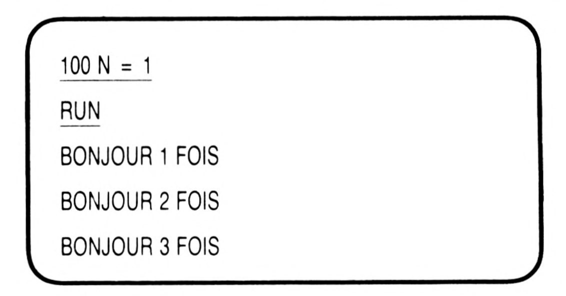
Utiliser le compteur dans la boucle

Voyez ce programme :



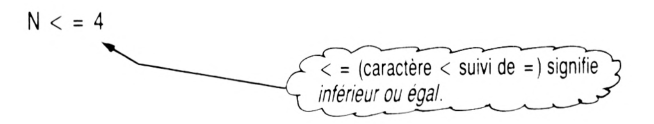
```
NEW
100 N = 0
110 PRINT "BONJOUR"; N; "FOIS"
120 N = N + 1
130 IF N < 4 THEN GOTO 110
RUN
BONJOUR 0 FOIS
BONJOUR 1 FOIS
BONJOUR 2 FOIS
BONJOUR 3 FOIS
```


En quelque sorte, il «numérote» chaque «BONJOUR» qu'il nous écrit. Mais, peut-être aimeriez-vous que le premier BONJOUR porte le numéro 1. Facile, direz-vous, il suffit de donner au départ la valeur 1 à N. Essayons :



```
100 N = 1
RUN
BONJOUR 1 FOIS
BONJOUR 2 FOIS
BONJOUR 3 FOIS
```

Cette fois on commence bien à 1, mais on s'arrête à 3 (comme tout à l'heure). Ce qui ne nous fait plus que 3 «BONJOUR» en tout. C'est bien normal puisque l'instruction 130 ne revient en 110 que si la valeur de N est *inférieure à 4*. Pour obtenir 4 fois «BONJOUR» (en partant toujours de 1), il faut remplacer la condition N < 4 par N < 5 ou, ce qui paraît plus naturel par :



```
N < = 4
```

Voici le programme définitif :

```
100 N = 1
110 PRINT "BONJOUR"; N; "FOIS"
120 N = N + 1
130 IF N < = 4 THEN GOTO 110
RUN
BONJOUR 1 FOIS
BONJOUR 2 FOIS
BONJOUR 3 FOIS
BONJOUR 4 FOIS
```



EXERCICES

1. Des deux instructions : IF et GOTO, laquelle est
   — un branchement inconditionnel ?
   — un branchement conditionnel ?

2. Que font chacun de ces programmes :

```
100 PRINT "SALUT"
110 PRINT "MON AMI"
120 GOTO 100
```

```
100 PRINT "SALUT"
110 PRINT "MON AMI"
120 GOTO 110
```

3. Quelle est l'instruction qui permet d'augmenter de 1 la valeur d'une variable I ?

4. Que fourniront chacun de ces programmes :

```
100 N = 0
110 PRINT "******"
120 N = N + 1
130 IF N < = 3 THEN GOTO 110
```

```
100 N = 1
110 PRINT "******"
120 N = N + 1
130 IF N < 3 THEN GOTO 110
```

5. Que signifient ces messages :
   a) Type mismatch error in 150
   b) Line does not exist in 200

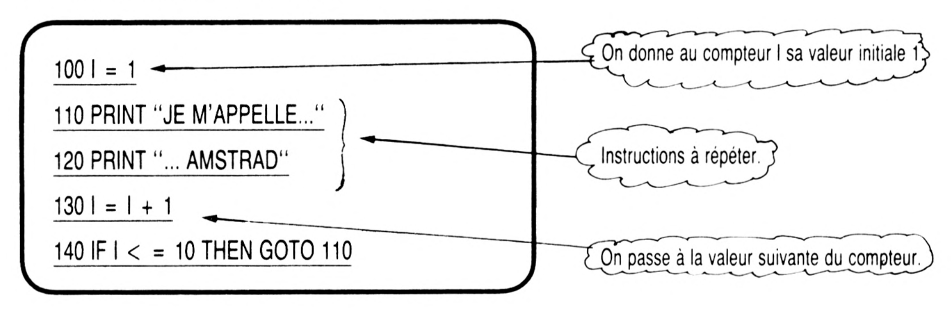
6. Comment interrompre un programme :
   a) lorsqu'il attend une réponse à INPUT
   b) dans les autres cas.

VI. Pour faire des boucles sans compter

C'était au temps où l'on comptait les tours

Nous avons vu comment répéter des instructions en comptant le nombre de tours de boucle à l'aide d'un compteur.

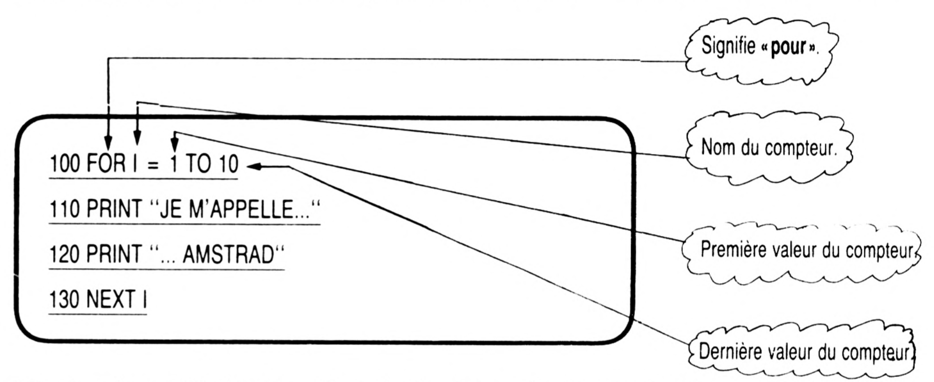
Par exemple, dans ce programme :



On exécute 10 fois les instructions 110 et 120. La première fois, le compteur I vaut 1 ; la seconde fois, il vaut 2, et ainsi de suite jusqu'à 10.

La boucle automatique

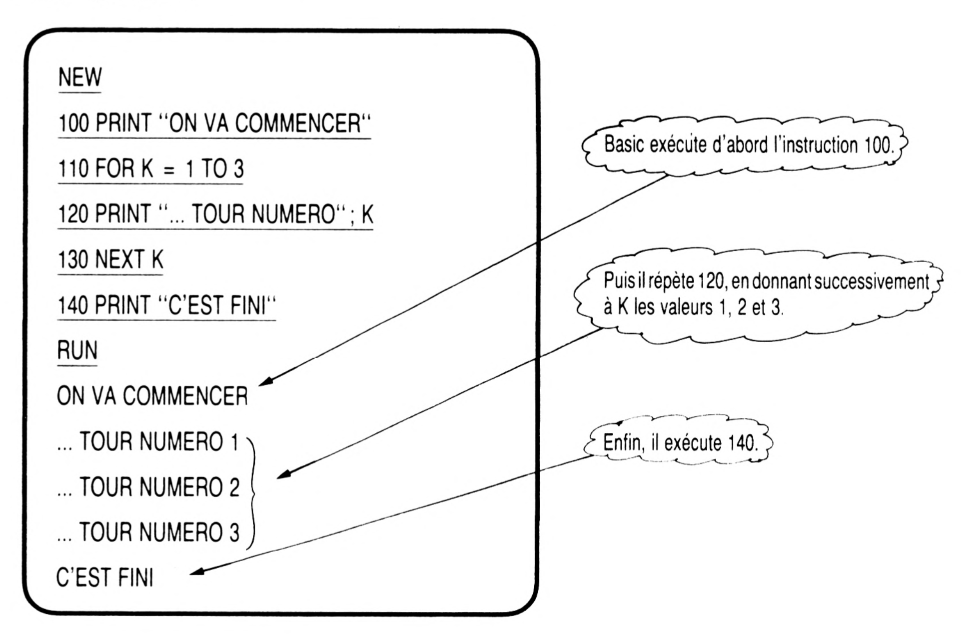
Basic vous offre une manière plus simple d'écrire le programme précédent :



L'instruction 100 signifie « Pour I allant de 1 à 10 ». Elle indique le début d'une « boucle » ; elle donne le nom du compteur et elle fixe ses valeurs de début et de fin. L'instruction 130 signale à Basic qu'il doit passer à la valeur suivante de I. Si celle-ci ne dépasse pas la limite fixée par l'instruction FOR (ici 10), Basic reprend l'exécution des instructions de la boucle (110 et 120).

Pour bien voir ce qui se passe

Voici un autre programme qui vous montre le fonctionnement des instructions FOR et NEXT :



Attention à bien placer le NEXT

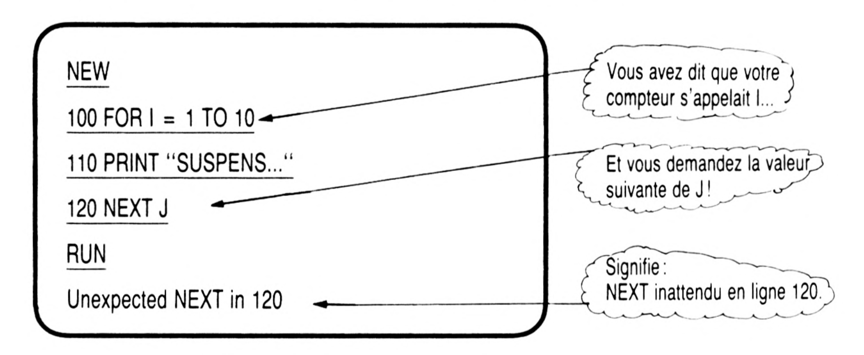
C'est l'instruction NEXT qui précise *jusqu'où* s'étend la boucle. Comparez ces deux programmes :

```
100 FOR I = 1 TO 3
110 PRINT "BONJOUR";
120 PRINT "MONSIEUR"
130 NEXT I
RUN
BONJOUR MONSIEUR
BONJOUR MONSIEUR
BONJOUR MONSIEUR
```

```
100 FOR I = 1 TO 3
110 PRINT "BONJOUR"
120 NEXT I
130 PRINT "MONSIEUR"
RUN
BONJOUR
BONJOUR
BONJOUR
MONSIEUR
```

A chaque FOR son NEXT

Pour l'instant, nos programmes ne comportent qu'une seule boucle. C'est pourquoi vous ne voyez peut-être pas pourquoi Basic vous impose de donner le nom du compteur après le mot NEXT. Mais, même dans ce cas, il vous faut prendre garde à donner le même nom dans FOR et NEXT, sinon...



Faites-vous la main

* Essayez de prévoir ce que produit ce programme, puis vérifiez-le :

```
100 PRINT "QUEL EST VOTRE NOM";
110 INPUT N$
120 PRINT "HELLO"
130 FOR C = 1 TO 4
140 PRINT N$
150 NEXT C
160 PRINT "COMMENT ALLEZ-VOUS"
```

* Même question en remplaçant la ligne 140 par :

```
140 PRINT N$;
```

* Même question en ajoutant :

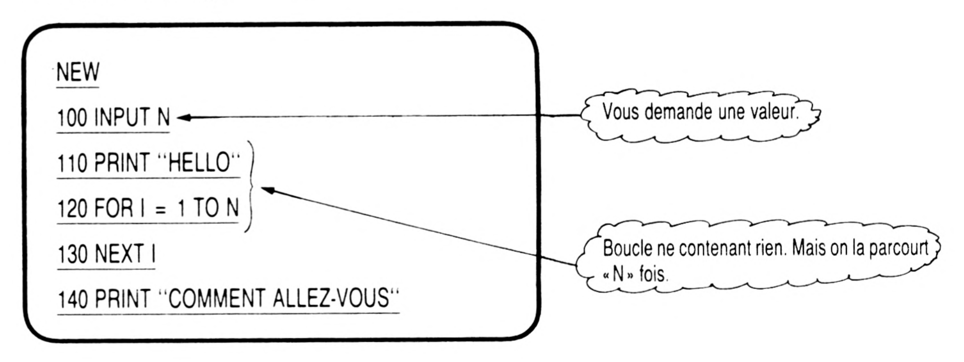
```
155 PRINT
```

* Même question avec ce programme :

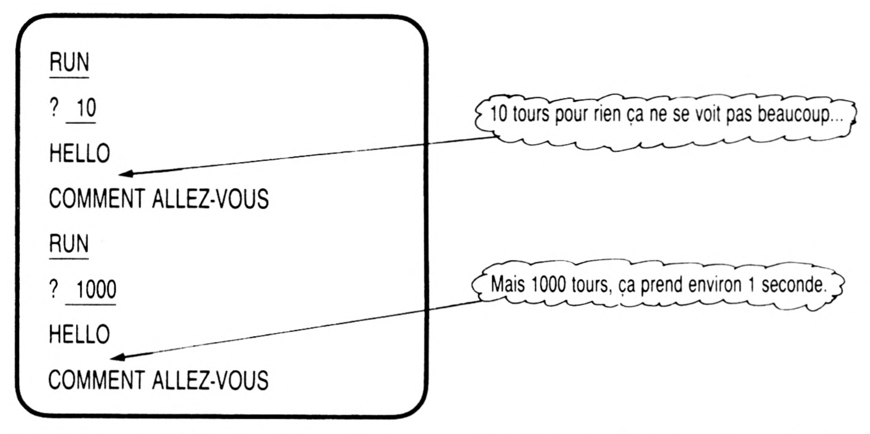
```
100 PRINT "TABLE DES 7"
110 FOR I = 1 TO 10
120 PRINT I; "FOIS 7 ="; I * 7
130 NEXT 1
```

Pour marquer un temps d'arrêt

Essayez ce programme :



avec différentes réponses :

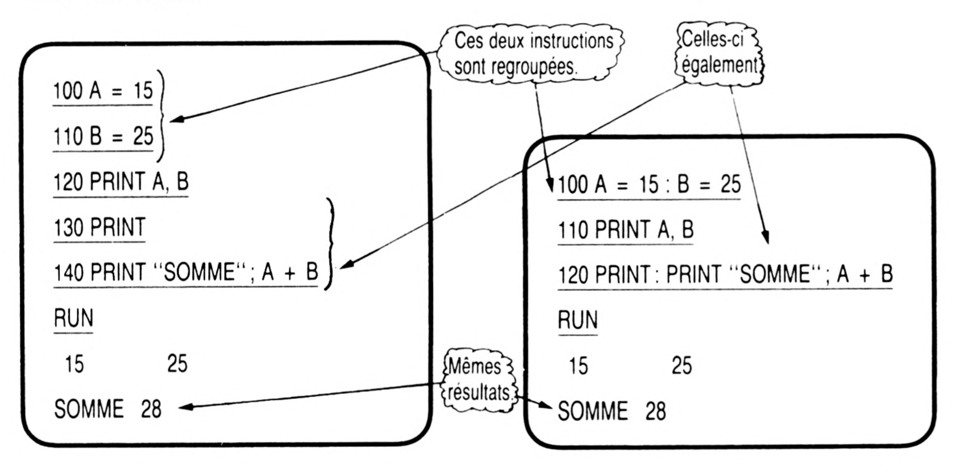


Bien sûr, une boucle vide comme celle formée par les instructions 120 et 130, cela semble inutile. Mais, il faut quand même un petit peu de temps à Basic pour la parcourir ; il faut bien qu'il ajoute 1 au compteur, qu'il l'examine et qu'il revienne en 100. Vous voyez que pour parcourir 1000 fois cette boucle, il faut environ une seconde.

A quoi bon faire des boucles vides? Tout simplement pour marquer un temps d'arrêt comme dans notre exemple. Dans certains cas, cela vous servira à ralentir quelque chose ; par exemple, dans un jeu, vous pourrez ainsi ralentir le déplacement d'un mobile,... Nous en rencontrerons des exemples par la suite.

Plusieurs instructions pour un même numéro

Jusqu'ici, pour faciliter la compréhension, nous n'avons placé qu'une instruction derrière un numéro. Votre Amstrad accepte que vous en placiez plusieurs en les séparant par deux points (:). Par exemple, ces deux programmes fournissent la même chose :



On dit parfois qu'on a placé plusieurs instructions sur une même « ligne ». Il s'agit d'une ligne pour Basic, mais pas obligatoirement d'une ligne d'écran. Nous avions déjà dit qu'une instruction Basic pouvait occuper plusieurs lignes d'écran (exactement deux). Vous voyez maintenant qu'une « ligne » d'un programme Basic peut comporter une ou plusieurs instructions (séparées par « : » et s'étendre sur une à six lignes d'écran.



Faites-vous la main

* Essayez de prévoir ce que produisent chacun de ces programmes. Vérifiez-le.

- 100 PRINT "VOTRE NOM": INPUT N$
  110 PRINT:PRINT
  120 PRINT "HELLO": PRINT N$

- 100 PRINT "VOTRE NOM": INPUT N$
  110 FOR I = 1 TO 1000: NEXT I
  120 PRINT: PRINT: PRINT
  130 PRINT "VOUS VOUS APPELEZ VRAIMENT"
  140 FOR I = 1 TO 2000: NEXT I
  150 PRINT: PRINT N$



EXERCICES

1. Ecrire un programme qui vous demande quelle table de multiplication vous voulez et qui vous l'affiche. Inspirez-vous de ce que vous avez vu dans le « faites-vous la main » de la page...
2. Réaliser un programme qui vous affiche les carrés des vingt premiers nombres entiers.
3. Ecrire un programme qui :
   — affiche « BONJOUR »
   — attend environ 1 seconde
   — affiche « UN INSTANT S.V.P. »
   — attend environ 5 secondes
   — efface l'écran
   — affiche « JE M'APPELLE AMSTRAD »
4. Que signifie le message :
   Unexpected NEXT in 220

VII. Pour prendre une décision :

Au chapitre V, nous avons appris à utiliser l'instruction IF pour réaliser des boucles. Ainsi, par exemple, nous avons employé :

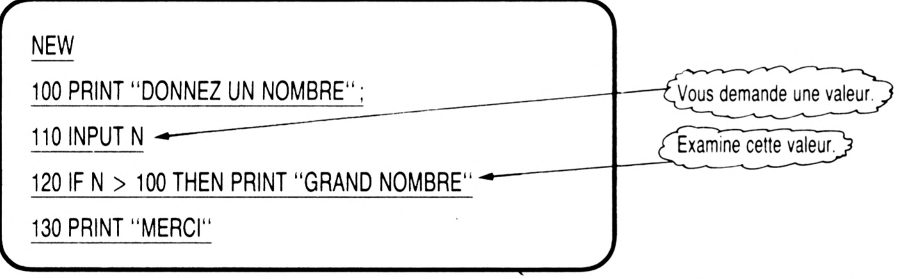
```
IF I < = 5 THEN GOTO 110
```

qui signifie : *si* la valeur de I est inférieur ou égale à 5, alors revenir en 110.

Mais nous allons voir qu'il existe d'autres utilisations de l'instruction IF.

Pour prendre une petite décision

Tapez ce programme :

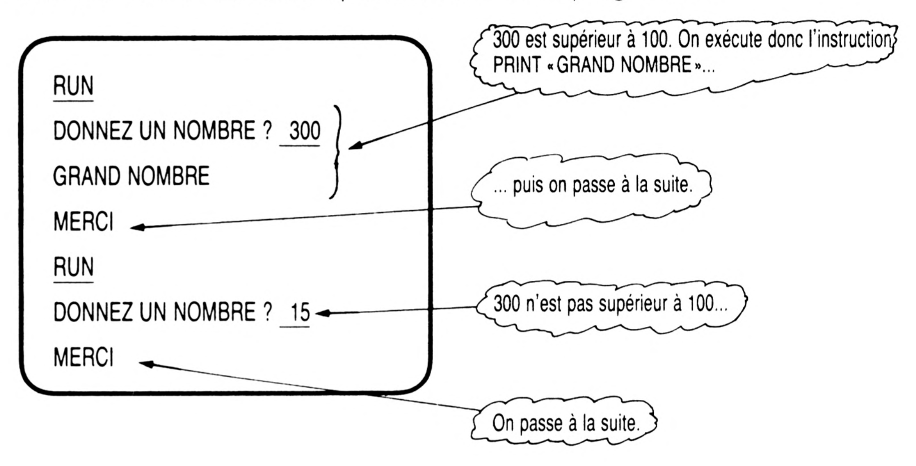


Cette fois, l'instruction 120 signifie : si la valeur de N est supérieure à 100, alors exécuter l'instruction :

```
PRINT "GRAND NOMBRE"
```

puis passer à la suite, c'est-à-dire la ligne 130. Si, par contre, la condition n'est pas vérifiée, passez immédiatement à l'instruction suivante.

Vérifions cela en exécutant plusieurs fois notre programme :



Voici un autre exemple où l'on compare, cette fois, des chaînes de caractères :

```
NEW
100 PRINT "QUEL EST VOTRE NOM"
110 INPUT N$
120 IF N$ = "LAURENCE" THEN PRINT "JOLI NOM"
130 PRINT "AU REVOIR"
RUN
QUEL EST VOTRE NOM
? PIERRE
AU REVOIR
RUN
QUEL EST VOTRE NOM
? LAURENCE
JOLI NOM
AU REVOIR
```

Si N$ contient la chaîne LAURENCE, on écrit JOLI NOM.

PIERRE est différent de LAURENCE... On passe à la suite.

Cette fois N$ contient LAURENCE. On écrit « JOLI NOM »...

... et on passe à la suite.

Faites-vous la main

* Essayez de prévoir ce qui fournira ce programme (vérifiez ensuite)

```
100 INPUT N1, N2
110 IF N1 < N2 THEN PRINT "DANS L'ORDRE"
```

suivant que vous lui répondez :
- 10, 15
- 25, 12
- 13, 13

* Essayez de prévoir ce que fournira ce programme (vérifiez)

```
100 INPUT N1, N2
110 IF N1 = 2 * N2 THEN PRINT "BRAVO"
```

suivant que vous lui répondez :
- 25, 15
- 80, 40
- 10, 20
- 0, 0
- 15, 15

* Essayez de prévoir ce que fournira ce programme :

```
100 PRINT "DITES OUI": INPUT R$
110 IF R$ = "O" THEN PRINT "MERCI"
```

suivant que vous répondez :
- OUI
- NON
- O

* Essayez de prévoir ce que fournira ce programme (< > signifie « différent de »)

```
100 PRINT "DITES OUI": INPUT R$
110 IF R$ <> "OUI" THEN PRINT "AH BON"
```

suivant que vous répondez :
- O
- NON
- OUI

Pour choisir entre deux possibilités

Essayez ce programme :

```
NEW
100 PRINT "DONNEZ UN NOMBRE";
110 INPUT N
120 IF N > 100 THEN PRINT "GRAND NOMBRE"
130 IF N < = 100 THEN PRINT "PETIT NOMBRE"
140 PRINT "MERCI"
RUN
DONNEZ UN NOMBRE ? 80
PETIT NOMBRE
MERCI
RUN
DONNEZ UN NOMBRE ? 120
GRAND NOMBRE
MERCI
```

Vous demande une valeur.

Ecrit « GRAND NOMBRE » si cette valeur est supérieure à 100.

Ecrit « PETIT NOMBRE » dans le cas contraire.



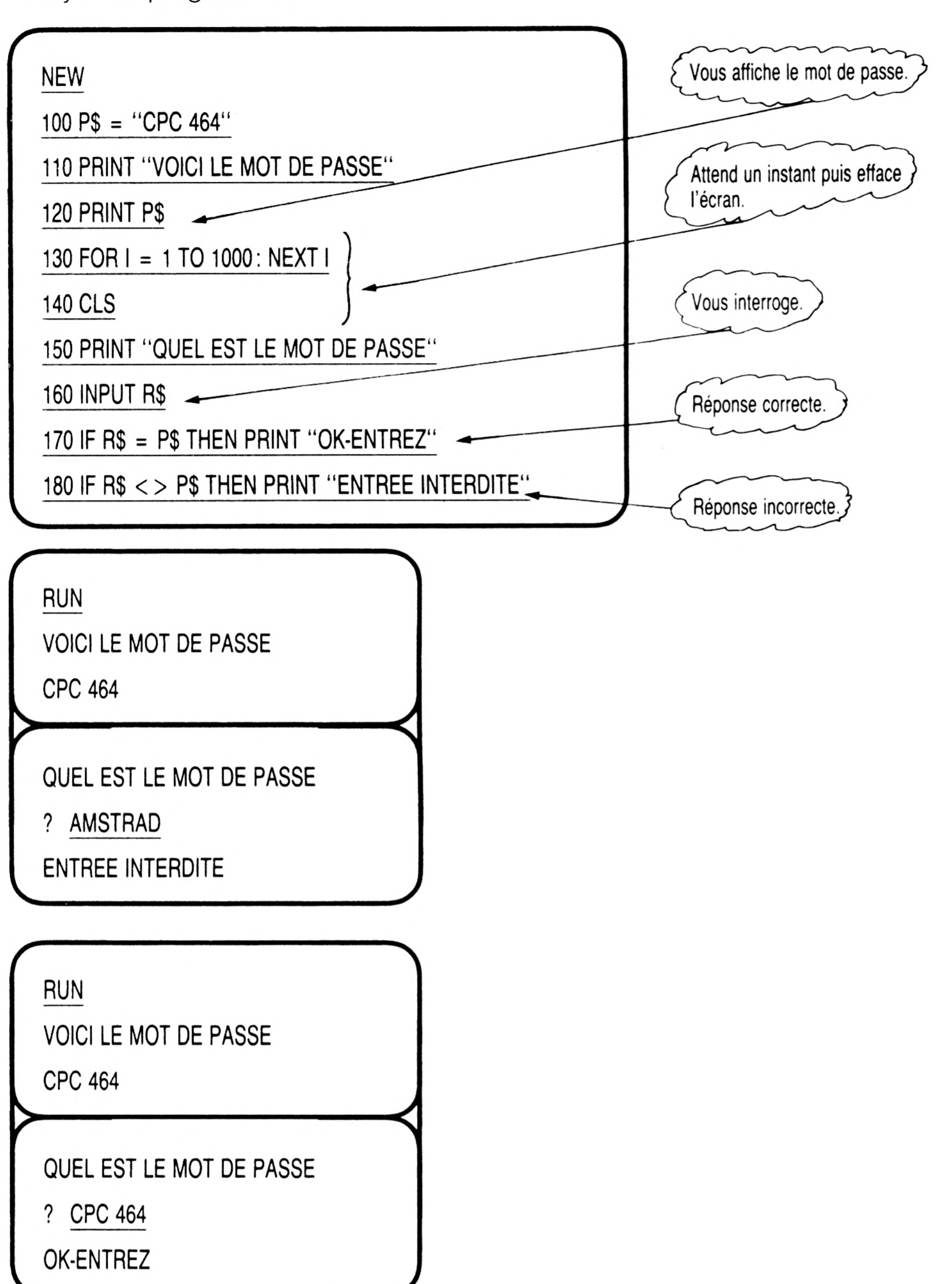
Faites-vous la main

* Dites ce que fournira le progamme précédent si vous répondez :
  • 300
  • 100
  Vérifiez.

* Mêmes questions si l'on remplace la ligne 130 par :
  130 IF N < 100 THEN PRINT "PETIT NOMBRE"

Connaissez-vous le mot de passe?

Essayez ce programme:



Pour faire des choix plus importants

L'instruction IF nous permet de choisir entre :

— exécuter une instruction, par exemple : écrire « GRAND NOMBRE »
— ne pas l'exécuter.

En combinant deux instructions IF, nous pouvons choisir entre *deux instructions*, par exemple :

— écrire « GRAND NOMBRE »
— écrire « PETIT NOMBRE »

Mais, comment choisir entre deux possibilités qui ne peuvent pas s'écrire en une seulę instruction ? Il faut alors combiner les instructions IF et GOTO. Nous en rencontrerons des exemples par la suite.

VIII. Jouer avec le hasard

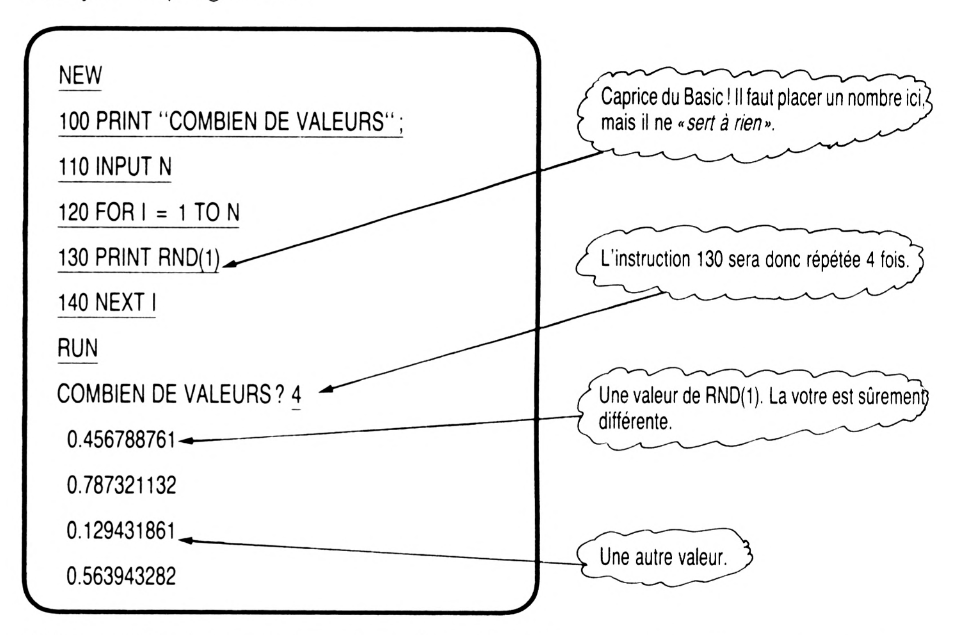


Beaucoup de jeux de société font intervenir le hasard : jeux de dés, de cartes, de roulette, etc. Votre Amstrad ne peut pas lancer un dé pour vous. Pourtant, il sait faire quelque chose de comparable : choisi au hasard l'un des nombres 1, 2, 3, 4, 5 ou 6.

En fait, vous verrez qu'en Basic, vous pourrez choisir beaucoup de choses au hasard : couleur, lettre, emplacement, son, etc. Et tout cela à l'aide d'un même mot Basic : RND.

Un mot magique : RND

Essayez ce programme :




```
NEW
100 PRINT "COMBIEN DE VALEURS";
110 INPUT N
120 FOR I = 1 TO N
130 PRINT RND(1)
140 NEXT I
RUN
COMBIEN DE VALEURS ? 4
 0.456788761
 0.787321132
 0.129431861
 0.563943282
```

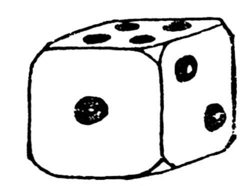
RND est l'abréviation du mot anglais « RaNDom » qui signifie « au hasard ».

RND(1) signifie : choisir au hasard un nombre décimal compris entre 0 et 1. La valeur exacte 1 ne peut jamais être obtenue.



Faites-vous la main

* Essayez le programme précédent avec de plus grandes valeurs pour N. Pour voir davantage de choses sur l'écran, ajoutez une virgule ou un point-virgule à la fin de l'instruction 130.
* Voyez ce qui se passe lorsque vous remplacez la valeur 1 de la ligne 130 par autre chose : 2, 5, 10, 50,...
* Voyez ce qui arrive si vous écrivez RND au lieu de RND(1).



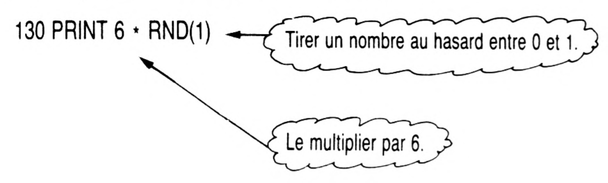
Pour faire rouler un dé

Comment transformer le programme précédent pour qu'il joue aux dés pour nous ? Autrement dit, comment faire pour qu'il nous fournisse l'un des nombres 1, 2, 3, 4, 5 ou 6 ?

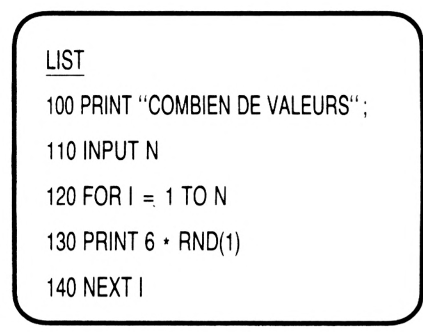
Pour l'instant, direz-vous, nous ne savons obtenir (avec RND) que des *nombres décimaux* compris *entre 0 et 1*. Nous allons progressivement modifier le programme précédent pour arriver au résultat voulu.

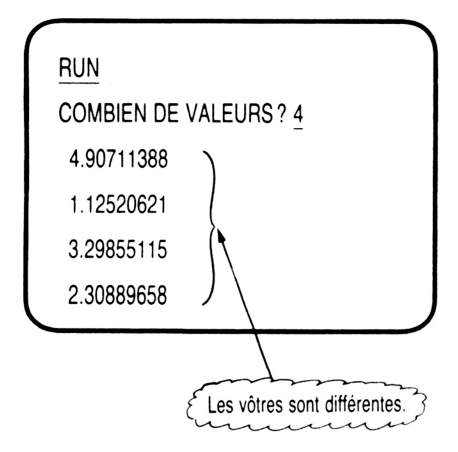
• *Multiplions par 6*

Si vous remplacez l'instruction 130 par :



vous obtenez alors des *nombres décimaux* compris *entre 0 et 6*. Vérifions-le :



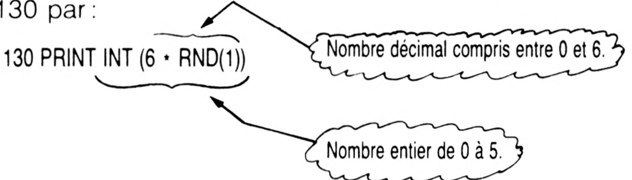


• *Supprimons les décimales*

Nous allons utiliser une nouvelle fonction Basic : INT. Ce nom est l'abréviation de INTeger qui signifie : « entier ». Voyez son rôle en essayant ces instructions en mode direct (ce qui vous évitera de détruire votre programme) :

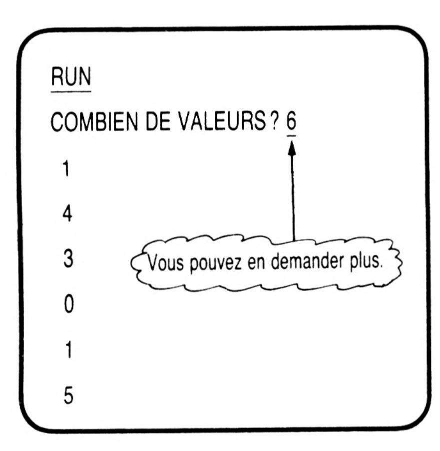
```
PRINT INT (4.253)
 4
PRINT INT (3.95)
 3
```

Appliquons donc cette fonction à la quantité RND(1). Pour cela, remplaçons la ligne 130 par :



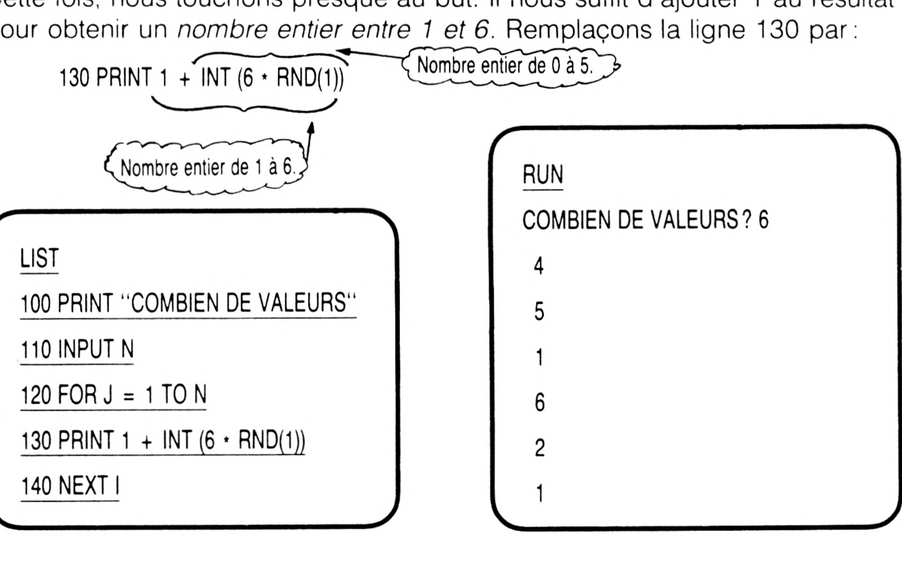
Nous obtenons ainsi un *nombre entier* compris *entre 0 et 5*. La valeur 6 ne pourra jamais être obtenue car, ne l'oubliez pas, RND(1) ne fournit jamais la valeur exacte 1. Vérifions tout cela :

```
LIST
100 PRINT "COMBIEN DE VALEURS";
110 INPUT N
120 FOR I = 1 TO N
130 PRINT INT (6 * RND(1))
140 NEXT I
```



• *Ajoutons 1*

Cette fois, nous touchons presque au but. Il nous suffit d'ajouter 1 au résultat pour obtenir un *nombre entier entre 1 et 6*. Remplaçons la ligne 130 par :



Faites-vous la main

* Essayez de modifier le programme précédent pour qu'il fournisse des nombres entiers entre 1 et 10, entre 1 et 50, etc.

* Essayez de prévoir ce que vous obtiendrez en modifiant ainsi la ligne 130 :
  130 PRINT 2 + INT (6 * RND(1))
  Vérifiez-le.

* Même chose avec :
  130 PRINT 10 + INT (90 * RND(1))

Pour jouer à pile ou face

Nous savons fabriquer des nombres au hasard. Comment faire un tirage pile ou face ? Il nous suffit d'attribuer une valeur à chaque cas. Par exemple, nous pouvons convenir que :

1 représente PILE
2 représente FACE



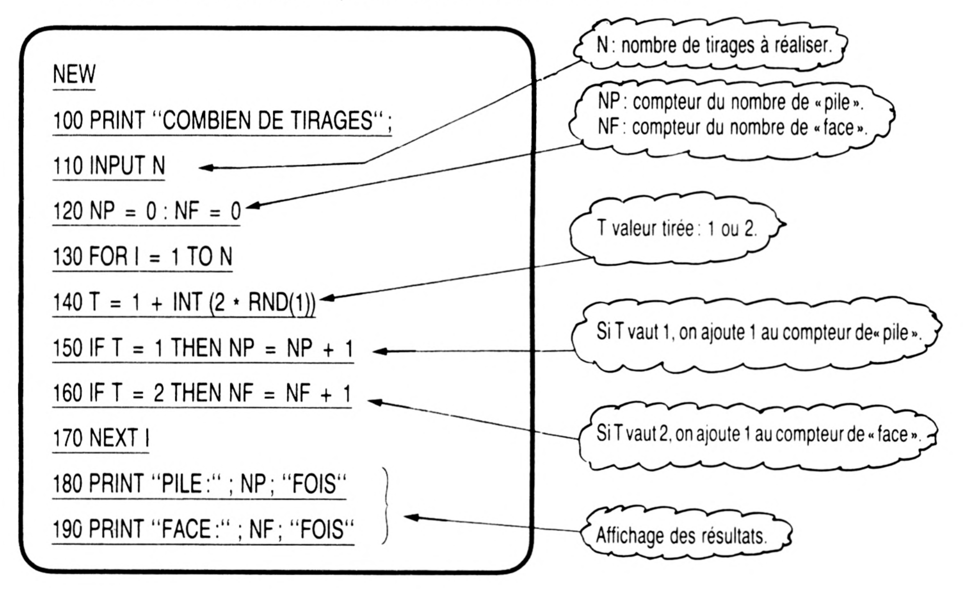


Tirer à pile ou face reviendra alors à tirer au hasard un nombre entier entre 1 et 2, ce que nous savons faire.

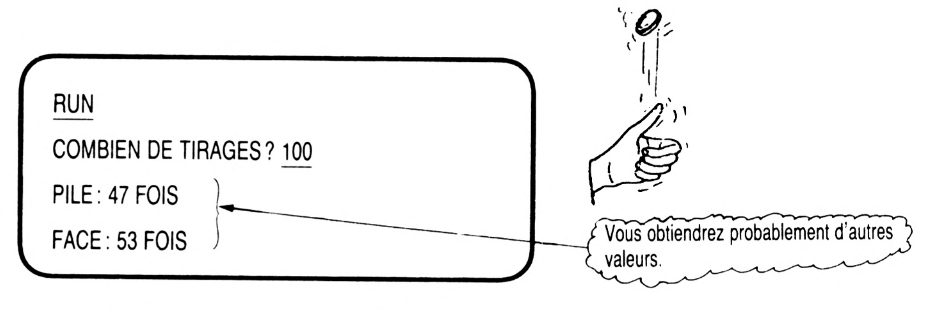
Pour compter les coups

Nous pouvons maintenant écrire un programme qui :

— nous laisse choisir le nombre de tirages
— compte le nombre de « pile » et le nombre de « face ».



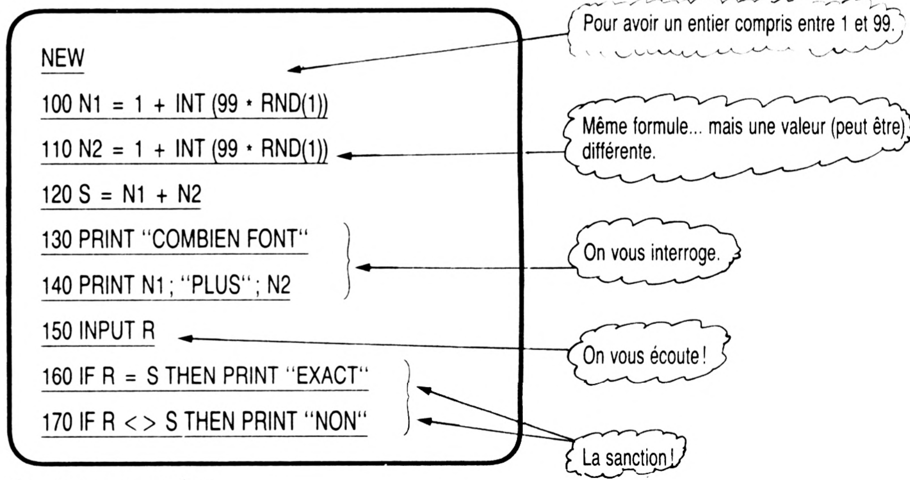
```
NEW
100 PRINT "COMBIEN DE TIRAGES";
110 INPUT N
120 NP = 0 : NF = 0
130 FOR I = 1 TO N
140 T = 1 + INT (2 * RND(1))
150 IF T = 1 THEN NP = NP + 1
160 IF T = 2 THEN NF = NF + 1
170 NEXT I
180 PRINT "PILE:" ; NP; "FOIS"
190 PRINT "FACE:" ; NF; "FOIS"
```



```
RUN
COMBIEN DE TIRAGES? 100
PILE: 47 FOIS
FACE: 53 FOIS
```

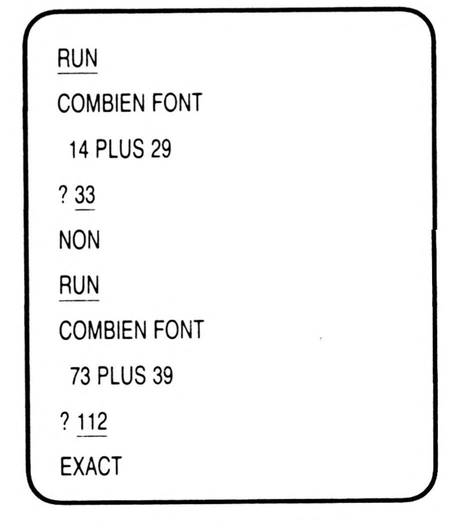

Pour s'entraîner au calcul mental

Voici un programme d'entraînement à la pratique de l'addition.

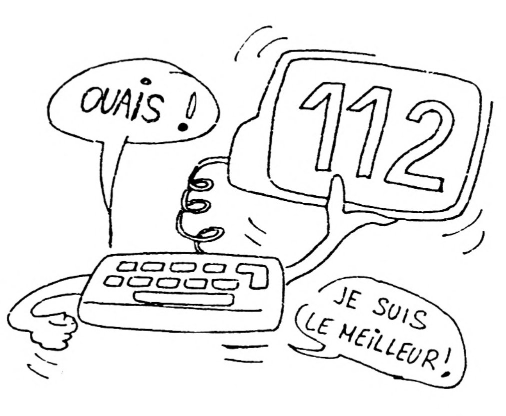


```
NEW
100 N1 = 1 + INT (99 * RND(1))
110 N2 = 1 + INT (99 * RND(1))
120 S = N1 + N2
130 PRINT "COMBIEN FONT"
140 PRINT N1; "PLUS"; N2
150 INPUT R
160 IF R = S THEN PRINT "EXACT"
170 IF R <> S THEN PRINT "NON"
```

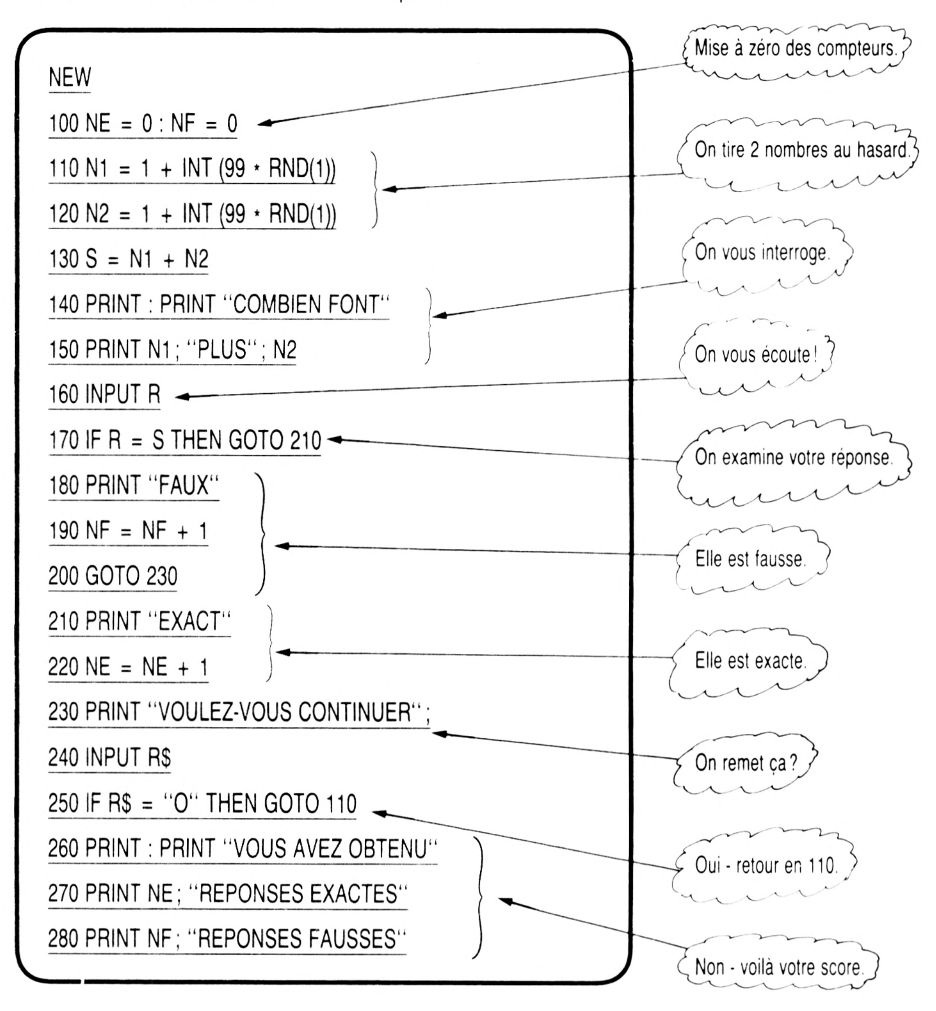

Quelques essais:



```
RUN
COMBIEN FONT
 14 PLUS 29
? 33
NON
RUN
COMBIEN FONT
 73 PLUS 39
? 112
EXACT
```



En fait, le programme ne propose qu'une seule opération à chaque fois. Faisons-le se poursuivre jusqu'à ce que vous souhaitiez vous arrêter. Comptons les « bonnes » et les « mauvaises » réponses :



```
NEW
100 NE = 0 : NF = 0
110 N1 = 1 + INT (99 * RND(1))
120 N2 = 1 + INT (99 * RND(1))
130 S = N1 + N2
140 PRINT : PRINT "COMBIEN FONT"
150 PRINT N1 ; "PLUS" ; N2
160 INPUT R
170 IF R = S THEN GOTO 210
180 PRINT "FAUX"
190 NF = NF + 1
200 GOTO 230
210 PRINT "EXACT"
220 NE = NE + 1
230 PRINT "VOULEZ-VOUS CONTINUER" ;
240 INPUT R$
250 IF R$ = "O" THEN GOTO 110
260 PRINT : PRINT "VOUS AVEZ OBTENU"
270 PRINT NE ; "REPONSES EXACTES"
280 PRINT NF ; "REPONSES FAUSSES"
```



Faites-vous la main

* Essayez de prévoir ce que fera le programme précédent si vous remplacez, en 110 et 120, la valeur 99 par 9. Vérifiez-le.
* Même question avec la valeur 999.
* Même question en modifiant ainsi les lignes 110 et 120 :

```
110 N1 = 10 + INT (90 * RND(1))
120 N2 = 10 + INT (90 * RND(1))
```

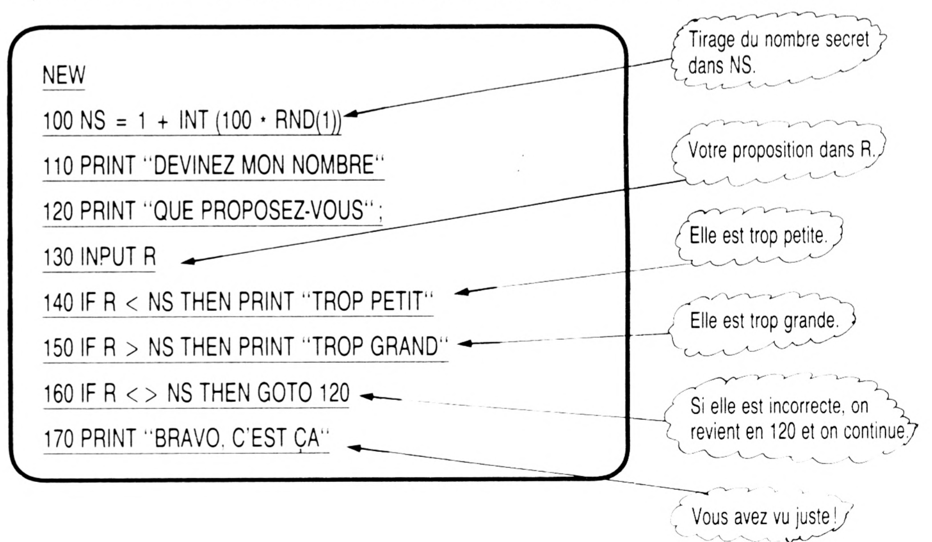
* Même question avec :

```
110 N1 = 100 + INT (900 * RND(1))
120 N2 = 10 + INT (90 * RND(1))
```

* Modifiez le programme pour qu'il vous propose des soustractions.
* Modifiez le programme pour qu'il vous propose des multiplications.

Le jeu du devin

Après le calcul mental, la récréation ! Voici un programme de jeu. Il choisit un nombre « en secret » et il vous demande de le deviner. Pour chacune de vos réponses, il vous précise si vous avez visé trop haut, trop bas ou juste !



Faisons un essai :

```
RUN
DEVINEZ MON NOMBRE
QUE PROPOSEZ-VOUS ? 50
TROP GRAND
QUE PROPOSEZ-VOUS ? 25
TROP PETIT
QUE PROPOSEZ-VOUS ? 37
BRAVO, C'EST ÇA
```

Vous aurez peut-être moins de chance que nous.

IX. Pour avoir du caractère

Une nouvelle façon de fabriquer des caractères

Nous avons appris à manipuler des caractères à l'aide de chaînes. Par exemple, pour écrire à l'écran le caractère B, nous pouvons utiliser ces instructions :

```
C$ = "B"
PRINT C$
```

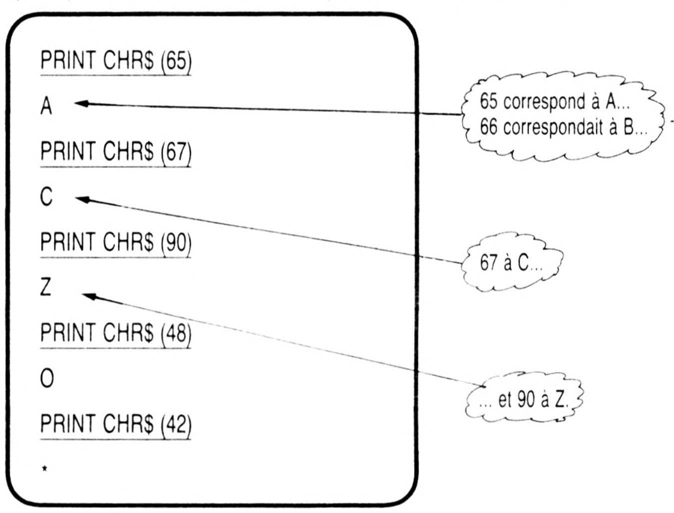
ou encore, plus simplement :

```
PRINT "B"
```

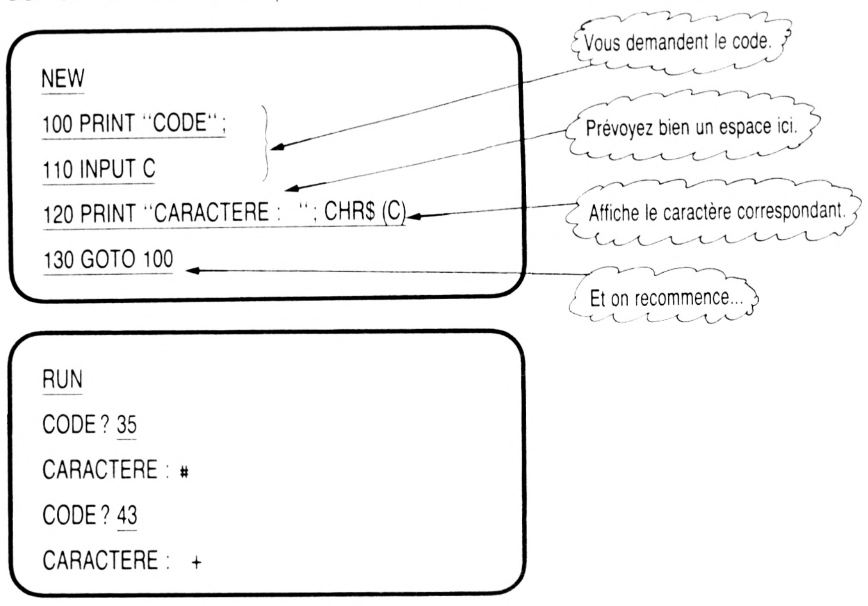
Mais, vous pouvez obtenir la même chose avec :

```
PRINT CHR$ (66)
B
```

CHR$ (66) signifie : le caractère ayant pour numéro 66. En fait, Basic attribue à chaque caractère, une valeur différente qu'on appelle son «*code*». Faisons quelques autres essais, toujours en mode direct :

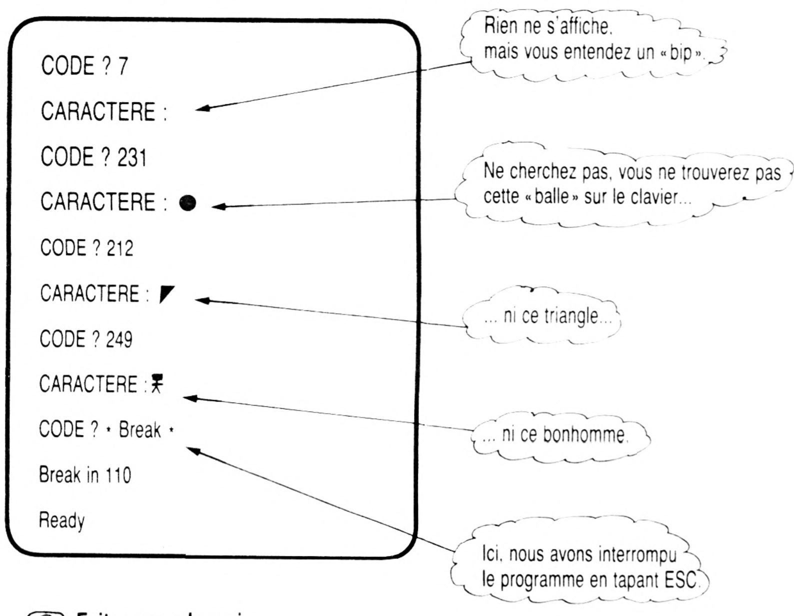


Pour en savoir un peu plus, nous pouvons écrire un programme qui nous demande un code et qui nous affiche le caractère correspondant :




```
RUN
CODE ? 35
CARACTERE : #
CODE ? 43
CARACTERE : +
```

Tout cela paraît assez naturel. Mais essayons d'autres valeurs :



Faites-vous la main

* Si vous êtes curieux, vous pouvez essayer d'autres valeurs de votre choix. Si vous rencontrez un problème, vous risquez d'être obligé de taper CTRL + SHIFT + ESC ; vous perdrez alors votre programme que vous devrez retaper.

* Voyez ce qui se passe quand vous fournissez une valeur supérieure à 255 ou négative.

Vous constatez qu'il existe 256 codes différents. La plupart d'entre eux (compris entre 32 et 255) correspondent à un véritable caractère que l'on peut afficher à l'écran. Certains d'entre eux (de 128 à 255) n'ont pas leur équivalent sur le clavier ; ils permettent de « dessiner » pas mal de choses. Enfin, les codes 0 à 31 ne correspondent pas à un caractère affichable ; ils réalisent une certaine fonction (bip, déplacement de curseur, ...)

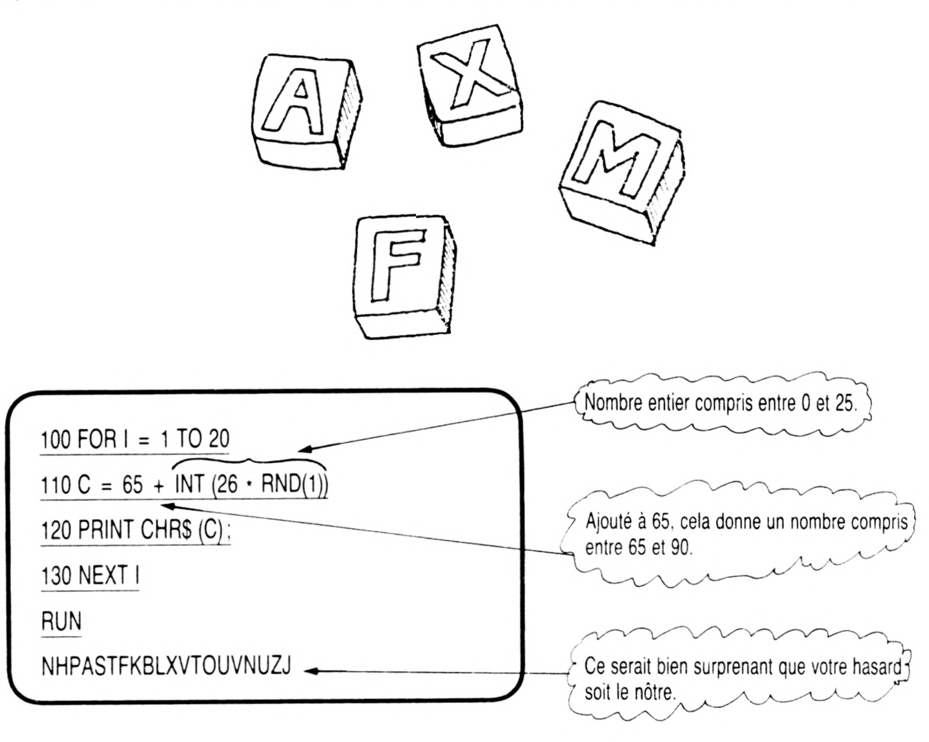
Pour tirer des lettres au hasard

Nous avons appris à tirer des nombres au hasard. Mais, dans certains programmes (de jeux, par exemple), vous aurez besoin de tirer des lettres au hasard. Nous allons voir comment y parvenir grâce à la fonction CHR$.

Pour cela, il vous suffit de savoir que les codes des vingt-six lettres A à Z vont de 65 à 90. Il est facile de le vérifier avec ce petit programme :

```
100 FOR I = 65 TO 90
110 PRINT CHR$ (I);
120 NEXT I
RUN
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Comment tirer une lettre au hasard. Il suffit de tirer son code au hasard. Cela revient à choisir un nombre entier compris entre 65 et 90. Voici un programme qui nous affiche une suite de 20 lettres tirées au hasard parmi l'alphabet :



Quand INPUT ne convient plus...

Jusqu'ici, votre seul moyen de communiquer une information au programme résidait dans l'instruction INPUT. Or, celle-ci n'est pas toujours pratique, notamment dans les jeux vidéo. En voici différentes raisons :

• Elle affiche un point d'interrogation à l'écran puis elle reproduit ce que vous tapez au clavier. Dans un jeu d'animation, cela risque fort d'être du plus mauvais effet.

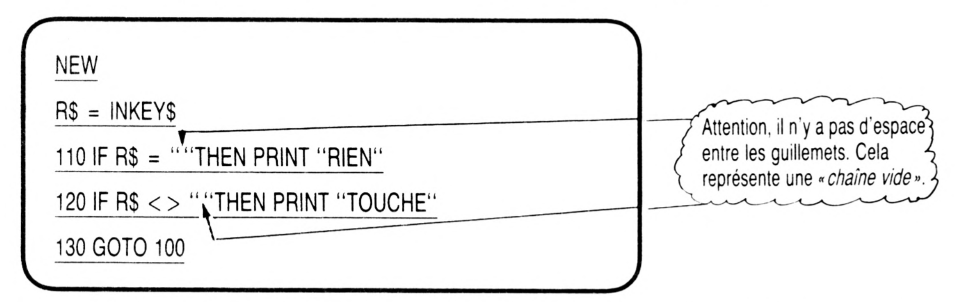
• Elle vous oblige à taper sur « RETURN » pour chaque information. Imaginez-vous en train de commander ainsi le déplacement d'un mobile sur l'écran : un petit déplacement à droite, RETURN, un petit déplacement à gauche, RETURN,... Pratique, non !

• Enfin, elle interrompt l'exécution du programme jusqu'à ce que vous ayez fourni votre réponse. Elle vous oblige donc toujours à taper quelque chose. Là encore, dans de nombreux jeux, vous avez besoin de savoir, par exemple, si le joueur a demandé un tir. Pour cela, vous examinez si une certaine touche (convenue à l'avance) a été pressée. Mais si elle ne l'est pas, vous ne passez surtout pas votre temps à attendre qu'elle le soit : sinon le jeu s'arrêterait jusqu'à ce que vous déclenchiez un tir !

Nous allons voir comment résoudre tous ces problèmes.

La fonction INKEY$

Elle va vous permettre de savoir si une touche du clavier est enfoncée. Elle vous dira même laquelle. Pour l'instant, essayez ce petit programme :



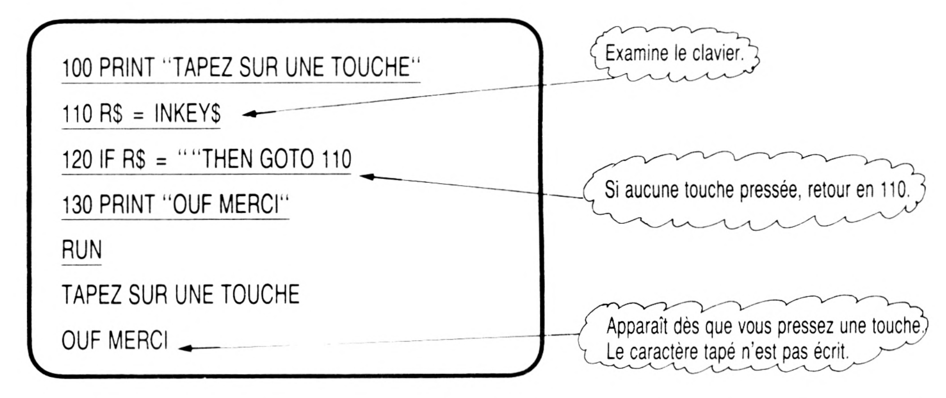
Il n'arrête pas d'afficher quelque chose à l'écran. Quand vous n'appuyez sur aucune touche, il affiche (sans cesse) « RIEN ». Dès que vous appuyez sur une touche, il écrit une fois « TOUCHE » puis recommence à afficher « RIEN ». Pour arrêter le programme, utilisez la touche habituelle ESC (2 fois).

En fait, l'instruction 100 examine si une nouvelle touche a été pressée. Si c'est le cas, elle place le caractère correspondant dans R$. Dans le cas contraire, elle y place la « chaîne vide », c'est-à-dire une chaîne ne contenant aucun caractère. Puis, sans attendre, Basic passe à la suite du programme. Les instructions 110 et 120 examinent le contenu de R$ et vous affichent, soit « RIEN », soit « TOUCHE ». Enfin, 130 fait recommencer le tout.

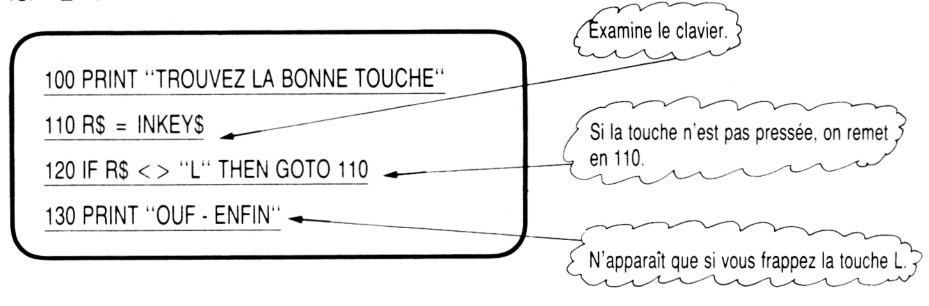
Remarques :

1. Si vous laissez une touche enfoncée pendant un moment, vous allez constater qu'elle est « prise en compte » plusieurs fois. Vous verrez apparaître « TOUCHE » à nouveau. Ceci provient de ce que la plupart des touches de l'AMSTRAD sont « à répétition automatique ». Vous l'avez peut-être déjà constaté lorsque vous frappez quelque chose et que vous laissez un peu « traîner » vos doigts sur les touches.
2. Le caractère détecté par INKEY$ n'est pas affiché.

Voici un autre exemple :



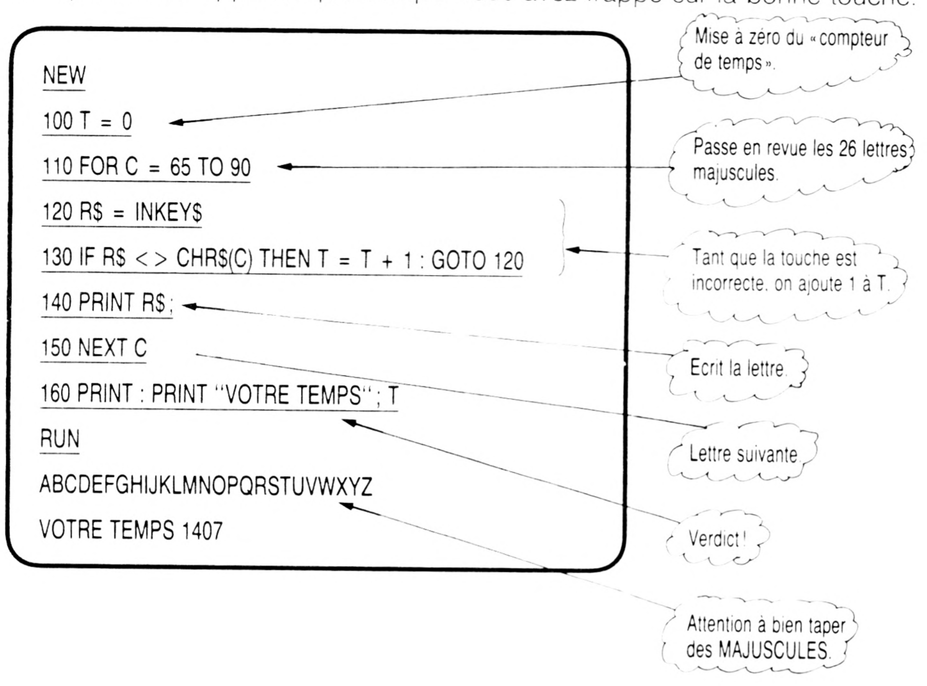
Et un dernier où l'on attend que vous appuyez sur une touche bien déterminée, ici « L » :



Et maintenant, place à la récréation avec deux jeux qui utilisent tout ce que nous venons d'apprendre dans ce chapitre.

Alphabet

Ce programme vous demande de frapper, dans l'ordre, les lettres de l'alphabet. Chaque lettre n'apparaît que lorsque vous avez frappé sur la bonne touche.



* Faites bien attention à la ligne 130. Elle signifie :

- Si la condition R$ < > CHR$(C) est vraie, alors on exécute les deux instructions placées après THEN :

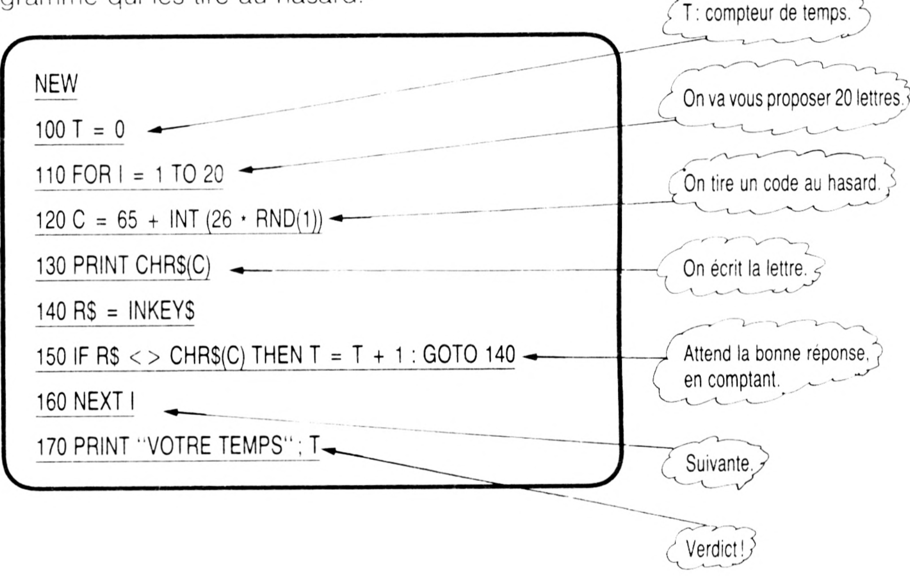
  T = T + 1
  GOTO 120 (ce qui nous ramène en 120).

- Si la condition est fausse, on passe à la ligne suivante : 140.

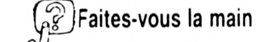
* Notez que T n'est pas un « vrai compteur » du temps écoulé. Il ne comptabilise ni des secondes, ni des minutes, mais seulement des tours de « boucle d'attente » : il mesure le nombre de fois que les instructions 120 et 130 ont été parcourues en attendant votre réponse.

Force de frappe

Dans le même ordre d'idée, voici un programme qui va vous permettre de vous familiariser avec le clavier. Lui aussi vous propose des lettres à taper le plus vite possible ; mais, cette fois, elles ne sont pas connues d'avance. C'est le programme qui les tire au hasard.




```
NEW
100 T = 0
110 FOR I = 1 TO 20
120 C = 65 + INT (26 * RND(1))
130 PRINT CHR$(C)
140 R$ = INKEY$
150 IF R$ <> CHR$(C) THEN T = T + 1 : GOTO 140
160 NEXT I
170 PRINT "VOTRE TEMPS"; T
```



Faites-vous la main

* Si vous préférez avoir des minuscules, remplacez la ligne 120 par :

120 C = 97 + INT (26 * RND (1))

X. Pour choisir vos couleurs



Jusqu'ici, nous avons laissé Basic écrire en jaune sur un fond bleu. Nous allons maintenant vous apprendre comment jouer avec les nombreuses couleurs de votre Amstrad. Si vous ne disposez que d'un moniteur monochrome, cela vous sera également utile ; simplement les différentes couleurs y apparaîtront sous forme de gris (ou de verts) plus ou moins brillants.

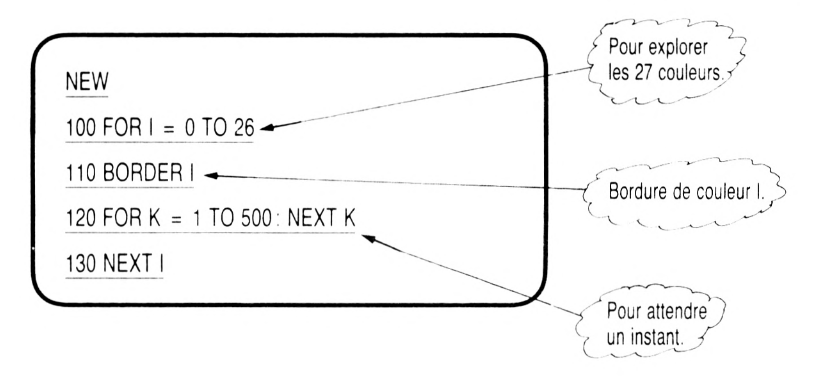
Pour distinguer le fond du pourtour

Vous avez certainement remarqué que Basic n'écrit que dans une partie de l'écran. Jusqu'ici, l'ensemble de l'écran était bleu foncé (à moins que vous n'ayez déjà essayé de jouer avec les couleurs !). Pour mieux distinguer la zone dans laquelle Basic écrit, essayez cette instruction :

```
BORDER 16
```

Vous constatez que le pourtour devient rose. (Si vous ne disposez que du noir et blanc, vous constaterez quand même un changement de gris.)

Votre Amstrad dispose de 27 couleurs. Chacune d'entre elles est repérée par un numéro allant de 0 à 26. Voici un programme qui les passe toutes en revue :





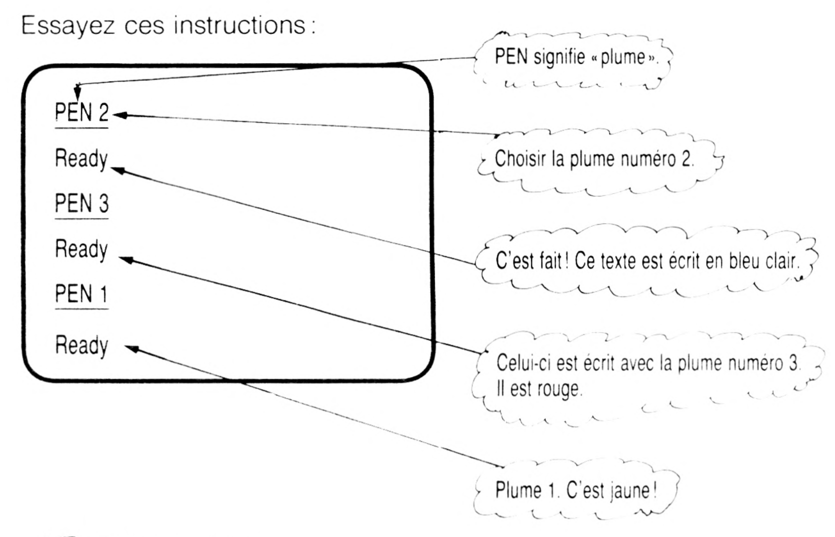
Faites-vous la main

* Modifiez le rythme de succession des couleurs en agissant sur le nombre 500 de la ligne 120.
* Pour vous donner la possibilité de contempler à loisir chacune des 27 couleurs, remplacez la ligne 120 par :
  120 R$ = INKEY$: IF R$ = " "THEN GOTO 120
  (le programme ne passera à la couleur suivante que lorsque vous taperez sur une touche).
* Pour connaître le numéro de la couleur, ajoutez la ligne :
  115 PRINT "BORDURE DE COULEUR"; I

Pour jouer avec la couleur du texte

Nous venons d'apprendre à choisir la couleur de la bordure. Mais, pour l'instant, le texte est toujours écrit en jaune sur fond bleu. Nous allons maintenant voir comment modifier la couleur du texte (le fond, ce sera pour bientôt. Patience, que diable !)

Essayez ces instructions :

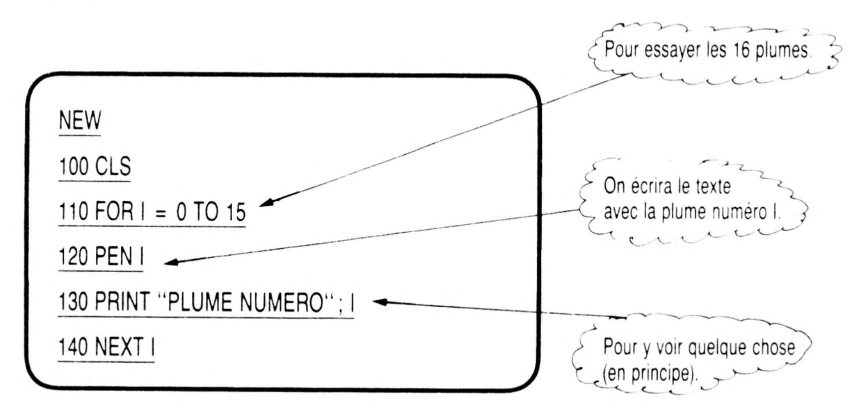




Faites-vous la main

* Essayez d'autres numéros de plume. Si vous rencontrez un problème, revenez à l'état initial en tapant CTRL + SHIFT + ESC.
* Voyez ce qui se passe avec la plume numéro 0.
* Voyez ce qui se passe avec des numéros négatifs ou supérieurs à 15. Le message obtenu signifie : « numéro incorrect ».

Mais, direz-vous, nous avons 27 couleurs et nous ne pouvons choisir que parmi 16 « plumes ». Et, pire que cela, vous devez avoir la vague impression que plusieurs plumes ont la même couleur ! Voici un programme qui va vous permettre d'y voir plus clair :



Si vous croyez qu'il y a des ratés, réfléchissez bien ! Lorsque la « plume » a la couleur du « fond », il est difficile de lire le texte. (Si vous n'êtes pas convaincu, essayez d'écrire sur une feuille rouge avec un crayon rouge !)

Finalement, vous obtenez en tout et pour tout quatre couleurs différentes. Ne vous laissez pas aller au découragement et ne reportez surtout pas votre Amstrad chez le vendeur ! Nous apprendrons plus tard à employer plus de couleurs.

Pour l'instant, nous pouvons dire que tout se passe comme si vous ne disposiez que de 4 plumes numérotées : 0, 1, 2 et 3. Quand vous choisissez un numéro supérieur à 3, Basic emploie quand même l'une de ces quatre plumes. Par exemple, si vous choisissez 4, il prendra 0 ; pour 5, il prendra 1, etc. (si vous êtes « matheux », vous reconnaissez la fonction « modulo 4 »).

Ne confondez pas numéro de plume et numéro de couleur

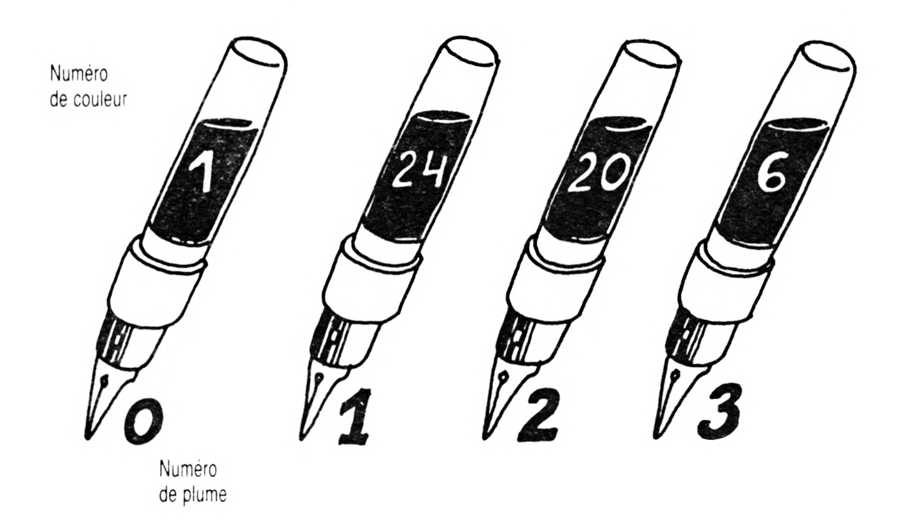
```
BORDER 1
```

signifie faire une bordure de couleur numéro 1 (bleu).

```
PEN 1
```

signifie : choisir la plume numéro 1. Cette plume écrit en jaune. Si vous cherchiez le numéro de couleur correspondant, vous trouveriez 24.

De la même façon, la plume 0 écrit avec la couleur 1, la plume 2 avec la couleur 20 et la plume 3 avec la couleur 6.



Pour jouer avec la couleur du fond

Revenez à l'état initial à l'aide des touches CTRL + SHIFT + ESC. Le fond redevient bleu et le texte redevient jaune.

Essayez alors ces instructions :

```
PAPER 2
Ready
PAPER 3
Ready
CLS

Ready
```

- PAPER signifie papier.
- PAPER 2 : Choisir le « papier » numéro 2.
- Ready : Ce texte est écrit en jaune sur fond bleu clair.
- Ready : Celui-ci est écrit sur un fond rouge.
- Ready : L'écran s'efface et devient rouge.

PEN permettait de choisir la couleur dans laquelle le texte doit être écrit. De la même manière, PAPER permet de choisir la couleur du fond sur lequel on va écrire.

L'instruction CLS efface l'écran, mais elle conserve le dernier « papier » choisi (ici 3). C'est pourquoi tout l'écran devient rouge.



Faites-vous la main

* Essayez l'instruction PAPER avec d'autres numéros. En cas de problème, revenez à l'état initial avec CTRL + SHIFT + ESC.

* Voyez ce qui se passe avec des numéros négatifs ou supérieurs à 15.

* Voyez ce qui se produit lorsque vous choisissez le même numéro pour le texte et pour le fond, par exemple :

```
PEN 3
PAPER 3
```

* Revenez à l'état initial en tapant CTRL + SHIFT + ESC. Tapez ces instructions :

```
PEN 1
PAPER 0
```

Vous constatez que les couleurs du texte et du fond ne changent pas. Cela signifie que lorsque vous allumez votre Amstrad ou lorsque vous revenez à l'état initial, Basic choisit la plume numéro 1 et le papier numéro 0 .

Pour changer les couleurs

Finalement, vous pouvez choisir pour la bordure n'importe laquelle des 27 couleurs (numérotées de 0 à 26). Par contre, pour le texte et le fond, tout se passe comme si vous ne disposiez que de 4 « stylos » (numérotés de 0 à 3).

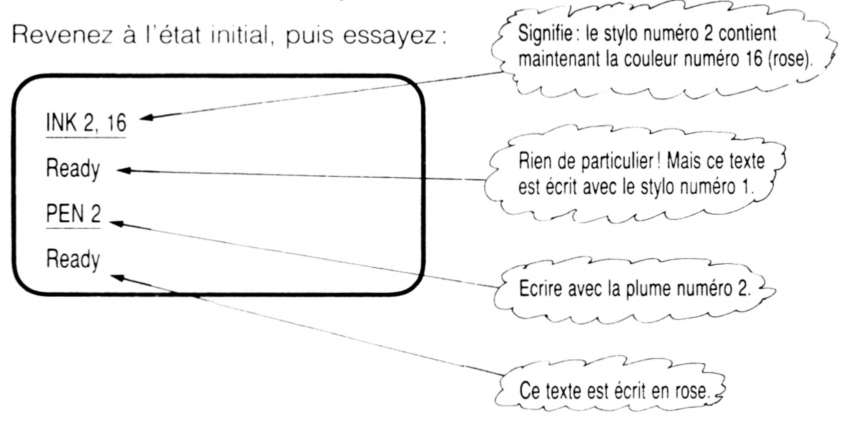
Au départ, votre Amstrad utilise le stylo numéro 1 pour le texte et le stylo numéro 0 pour le fond. Vous pouvez choisir un autre stylo pour le texte avec l'instruction PEN. De même, vous pouvez changer le stylo utilisé pour le fond avec l'instruction PAPER.

Mais les couleurs de ces 4 stylos semblent bien vous être imposées par l'Armstrad. Voici d'ailleurs la correspondance entre numéro de stylo et numéro de couleur :

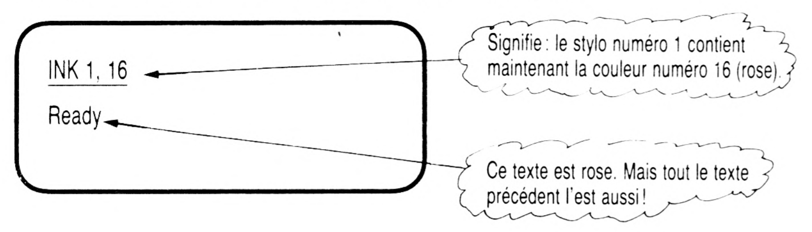
Numéro de « stylo »	Numéro de couleur	Couleur
0	1	bleu
1	24	jaune clair
2	20	bleu clair
3	6	rouge

Mais, en fait, vous pouvez modifier la couleur de chaque stylo grâce à une nouvelle instruction : INK. Celle-ci vous permet, en quelque sorte, de changer la « cartouche d'encre » des stylos.

Revenez à l'état initial, puis essayez :

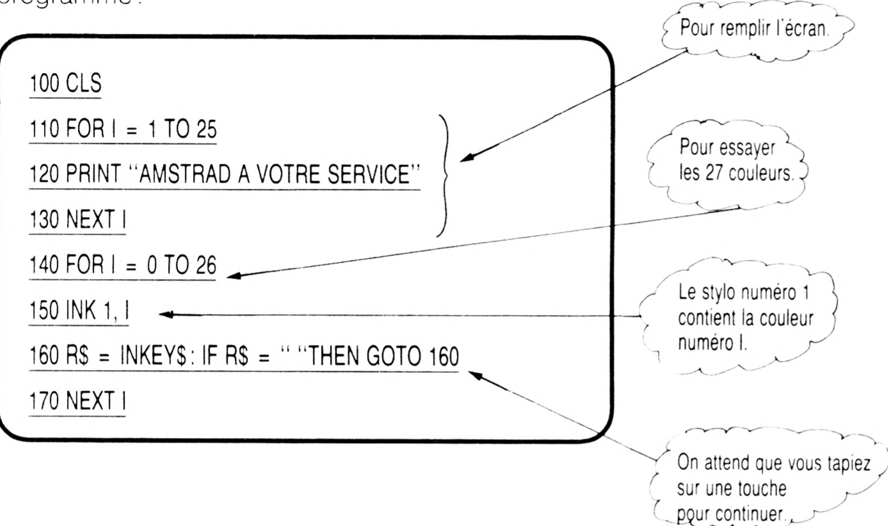


Revenez à nouveau à l'état initial, puis essayez :



Eh oui ! Quand vous changez la couleur d'un stylo, vous modifiez également la couleur de tout ce qui a déjà été écrit par ce stylo.

Revenez à l'état initial ; Basic redonne les cartouches initiales à chaque stylo. Ainsi, le stylo 1 contient maintenant la couleur habituelle : le jaune. Essayez ce programme :



REMARQUES :

1. Ici, nous n'utilisons que deux stylos à la fois : le 0 pour le fond et le 1 pour le texte. C'est pour cette raison que nous n'avons pas à employer d'instructions PEN ou PAPER.

2. Après avoir exécuté ce programme, la couleur d'encre n'est plus celle de l'état initial. Vous êtes alors peut-être tentés de taper CTRL + SHIFT + ESC. Cependant, si vous le faites, vous perdez votre programme ! Comment retrouver alors la couleur d'encre initiale ? Simplement en tapant (en mode direct) :

   INK 1, 24

Faites-vous la main

* Voyez comme le réglage de luminosité de votre téléviseur agit sur les couleurs en exécutant plusieurs fois le programme précédent.

* Modifiez ainsi l'instruction 150

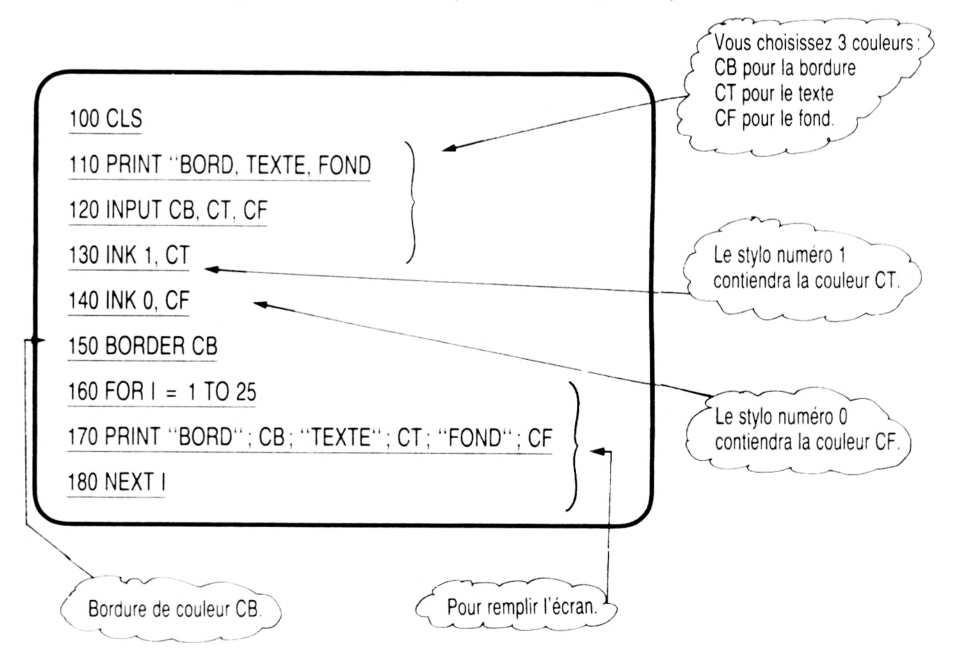
  150 INK 0, I

  Le programme ainsi obtenu vous permet de donner au fond successivement chacune des 27 couleurs. Pour retrouver la couleur de fond initiale, tapez (en mode direct) :

  INK 0, 1

Pour mesurer vos effets

Associer les différentes couleurs est un art parfois difficile. Voici un programme qui vous permet de juger de l'effet produit par l'association de trois couleurs de votre choix : une pour le texte, une pour le fond, une pour la bordure.



Faites-vous la main

* Essayez le programme précédent avec différentes couleurs de votre choix.
Si votre texte devient invisible, évitez de taper CTRL + SHIFT + ESC. Vous retrouverez les couleurs initiales en tapant :

```
INK 1, 24
INK 0, 1
```

(Bien sûr, il vous faudra taper la première instruction « à l'aveuglette » ; mais, si vous n'avez pas fait de faute, tout deviendra plus clair dès que vous appuyerez sur ENTER).

XI. Choisir son emplacement :

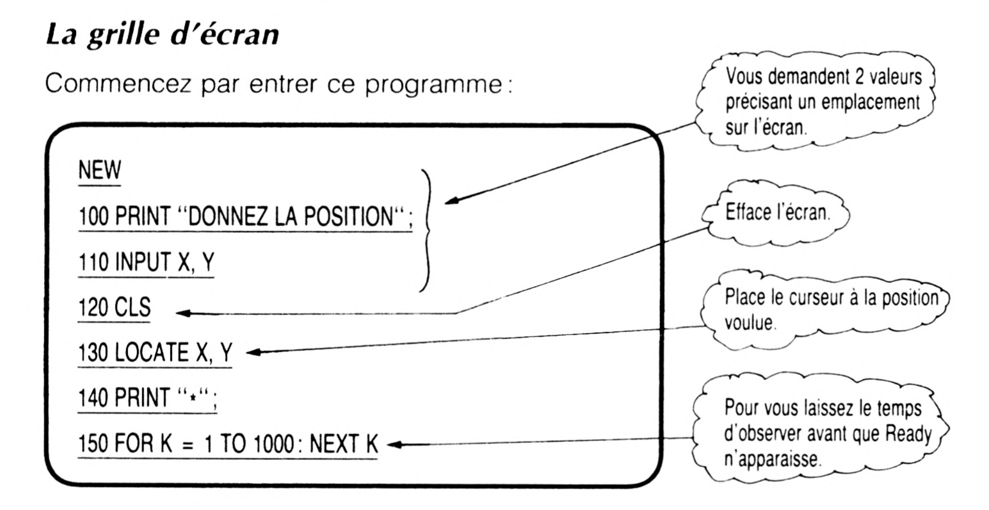


Jusqu'ici, nous nous sommes contentés d'écrire à l'écran ligne après ligne. Chaque nouvelle ligne « poussait » l'ensemble vers le haut, en faisant du même coup disparaître la première ligne. Cela ne présentait d'ailleurs aucun inconvénient dans les programmes que nous avons présentés.

Mais, comment, dans ces conditions, faire un dessin ? Mieux encore, comment faire se déplacer un objet sur l'écran (on dit aussi «*animer*» un *mobile*) ? Nous allons progressivement vous entraîner à cet aspect passionnant de la programmation qui constitue la base des «jeux vidéo».

La grille d'écran

Commencez par entrer ce programme :



```
NEW
100 PRINT "DONNEZ LA POSITION";
110 INPUT X, Y
120 CLS
130 LOCATE X, Y
140 PRINT "*";
150 FOR K = 1 TO 1000: NEXT K
```



Faites-vous la main

* Donnez d'abord à la bordure une couleur différente du fond avec l'instruction BORDER. Exécutez plusieurs fois ce programme en répondant :

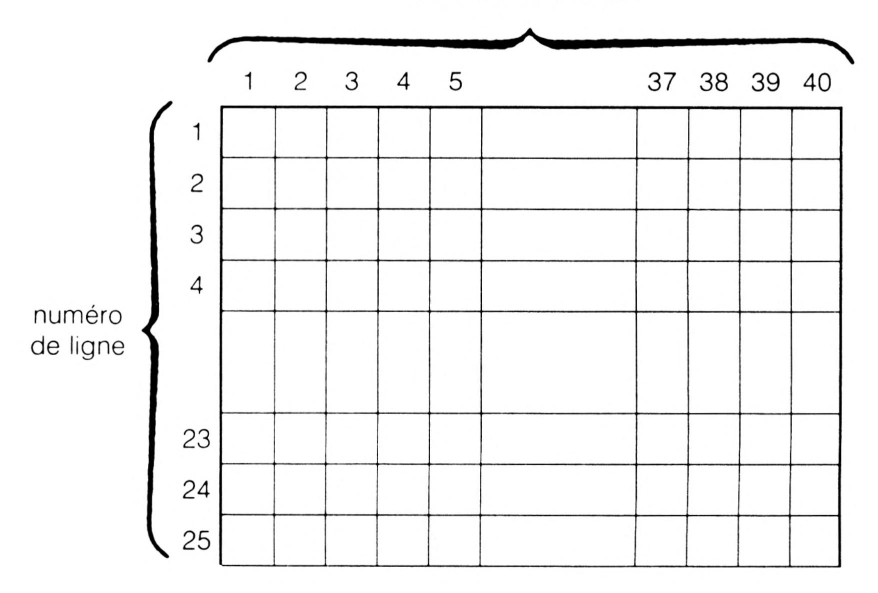
- 5,5
- 5,10
- 5,15
- 20,10
- 20,15
- 1,1
- 2,2
- 12,22
- 12,23
- 12,24
- 12,25
- 30,22
- 30,23
- 30,24
- 30,25
- 40,23
- 40,24
- 40,25

* Essayez d'autres valeurs de votre choix. Si, par malheur, vous « plantez » votre ordinateur, il se peut que vous soyez obligés de revenir à l'état initial.

Vous voyez qu'avec deux valeurs, vous pouvez déterminer la position exacte où va s'afficher l'étoile.

Comme vous l'avez probablement déjà constaté, votre écran peut afficher 25 lignes. Elles sont numérotées de 1 à 25, en partant du haut. Chaque ligne comporte 40 positions, numérotées de 1 à 40. Ainsi chaque emplacement de l'écran est repéré par deux valeurs qu'on appelle ses coordonnées :

— un numéro de ligne, compris entre 1 et 25
— un numéro dit « de colonne », compris entre 1 et 40



Dans le programme précédent, X contenait un numéro de colonne et Y contenait un numéro de ligne. L'instruction 120 permet de placer le curseur d'écran à la position indiquée. Ce curseur (invisible pendant le déroulement d'un programme) indique à quel endroit commencera le prochain PRINT.

Voilà donc pourquoi vous pouvez obtenir une étoile à un emplacement quelconque de l'écran.

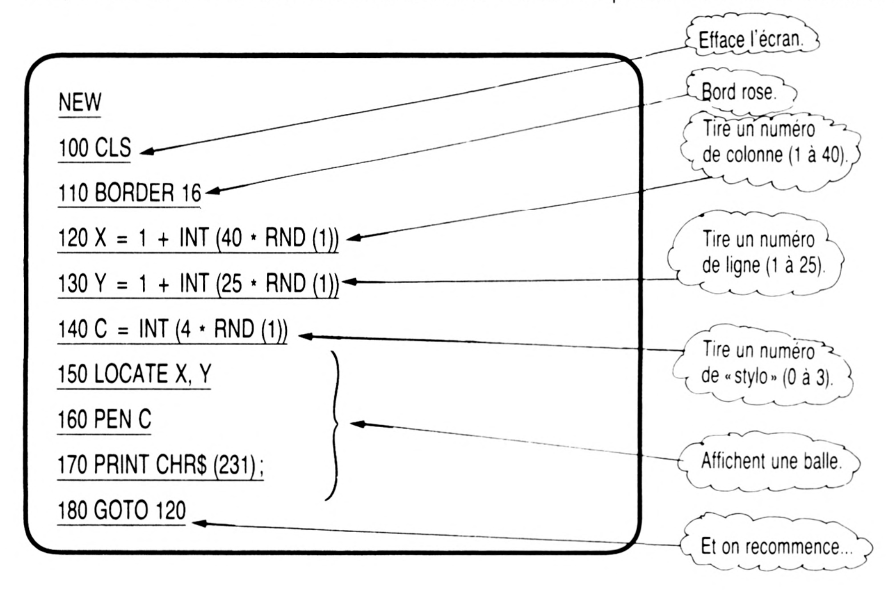


Faites-vous la main

* Remplacez la ligne 140 par :
  140 PRINT "BONJOUR"
  et exécutez le programme en fournissant les mêmes réponses qu'en page 110. Essayez ensuite :
  - 35, 10
  - 36, 10
  - 39, 10
* Affichez d'autres caractères que l'étoile en remplaçant successivement la ligne 140 par :
  140 PRINT CHR$ (224);
  140 PRINT CHR$ (229);
  140 PRINT CHR$ (238);
  140 PRINT CHR$ (239);
  (N'oubliez pas que, par exemple, CHR$ (224) signifie : le caractère ayant pour code 224).

Mosaïque sur écran

Voici un programme qui n'arrête pas d'afficher des ronds de couleur. A chaque fois, il choisit au hasard les coordonnées d'un emplacement et la couleur.



Comme de temps en temps, la couleur du rond est celle du fond, vous voyez certains ronds disparaître.

Pour arrêter le programme, il suffit de presser deux fois la touche ESC. Notez bien qu'il se peut que le message Ready ne soit pas visible. Dans ce cas, il vous suffit de retrouver la plume initiale en tapant (à « l'aveuglette ») :

```
PEN 1
```



Faites-vous la main

* Affichez d'autres caractères qu'une balle, en agissant sur la ligne 170. Essayez, par exemple, à la place de 231, les valeurs suivantes :
  - 220 • 203 • 221 • 224
  - 230 • 206 • 207 • 248

* Au lieu de faire afficher toujours le même caractère, faites-le tirer au hasard. Pour cela, ajoutez :
  165 CA = 128 + INT (128 * RND (1))
  (qui tire un code compris entre 128 et 255)
  et modifiez ainsi la ligne 170 :
  170 PRINT CHR$ (CA);
  Le résultat sera du meilleur effet.

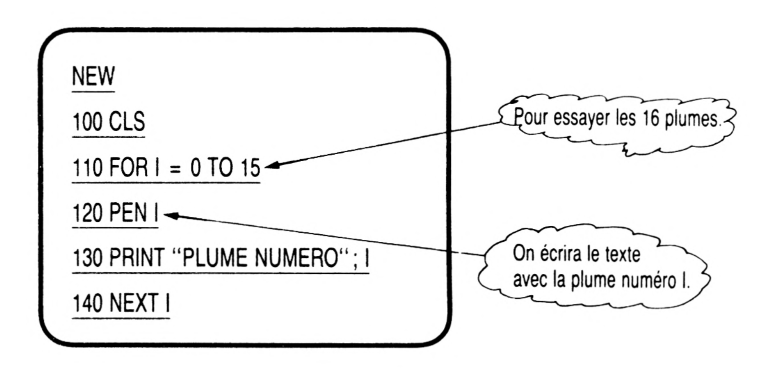
* Modifiez à votre gré les 4 couleurs de chacun des 4 « stylos » en ajoutant des instructions INK. Par exemple :
  112 INK 0,0
  114 INK 1,8
  116 INK 2,15
  118 INK 3,18
  En cas de problème, évitez de taper CTRL + SHIFT + ESC qui vous ferait perdre votre programme. Pour retrouver les couleurs initiales, il vous suffit de taper (en mode direct) :
  INK 0,1
  INK 1,24
  INK 2,20
  INK 3,6

Pour en voir de (presque) toutes les couleurs

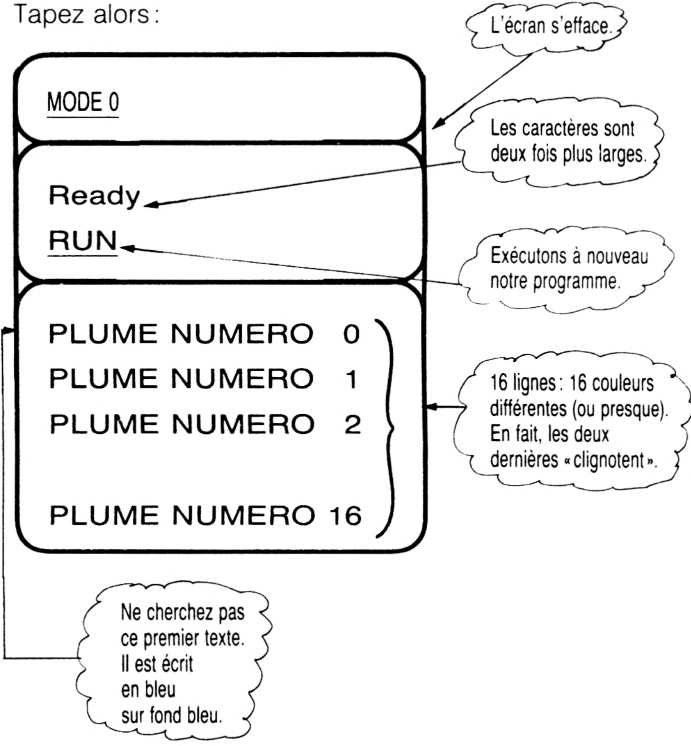
Bien sûr, grâce à INK, vous choisissez vos couleurs. Mais vous ne disposez malgré tout que de 4 stylos. C'est pourquoi vous ne pouvez pas utiliser plus de quatre couleurs à la fois.

Nous allons voir maintenant comment utiliser jusqu'à 16 stylos différents.

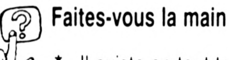
Commencez par revenir à l'état initial et à entrer à nouveau le programme que nous avions rencontré en page 101 :



Si vous l'essayez tel quel, vous n'obtiendrez bien sûr que 4 couleurs différentes. Tapez alors :



Vous voyez que l'instruction MODE 0 vous permet d'obtenir 16 couleurs différentes. Par contre, vous ne pouvez plus afficher que 20 caractères par ligne. (On n'a rien sans rien !)

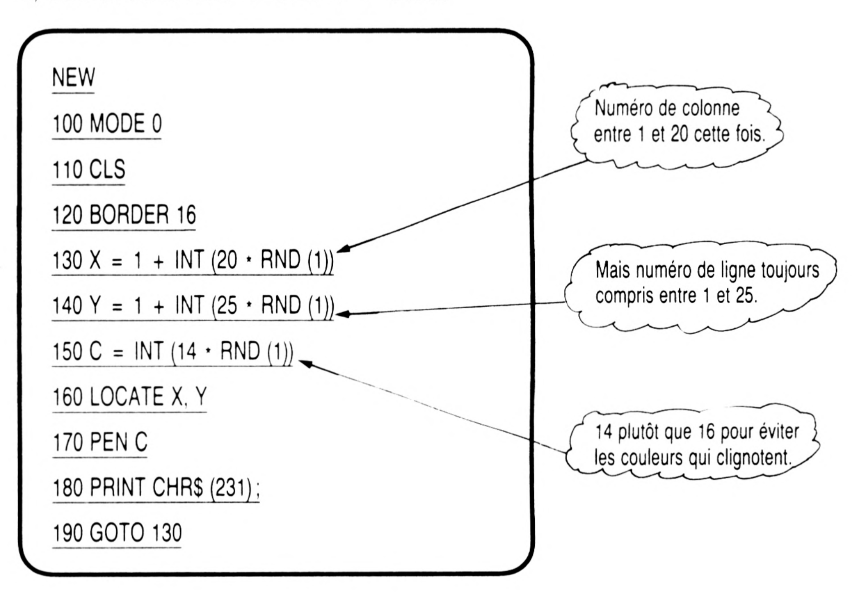


Faites-vous la main

* Il existe en tout trois « modes » sur votre Amstrad, numérotés 0, 1 et 2. Essayez chacun d'entre eux en faisant une liste de votre programme et en l'exécutant.

Super-mosaïque

Adaptons notre programme de mosaïque pour qu'il fonctionne en mode 0. Pour entrer des instructions, il vaut mieux revenir en mode 1, soit en tapant MODE 1, soit avec CTRL + SHIFT + ESC.



```
NEW
100 MODE 0
110 CLS
120 BORDER 16
130 X = 1 + INT (20 * RND (1))
140 Y = 1 + INT (25 * RND (1))
150 C = INT (14 * RND (1))
160 LOCATE X, Y
170 PEN C
180 PRINT CHR$ (231);
190 GOTO 130
```




Faites-vous la main

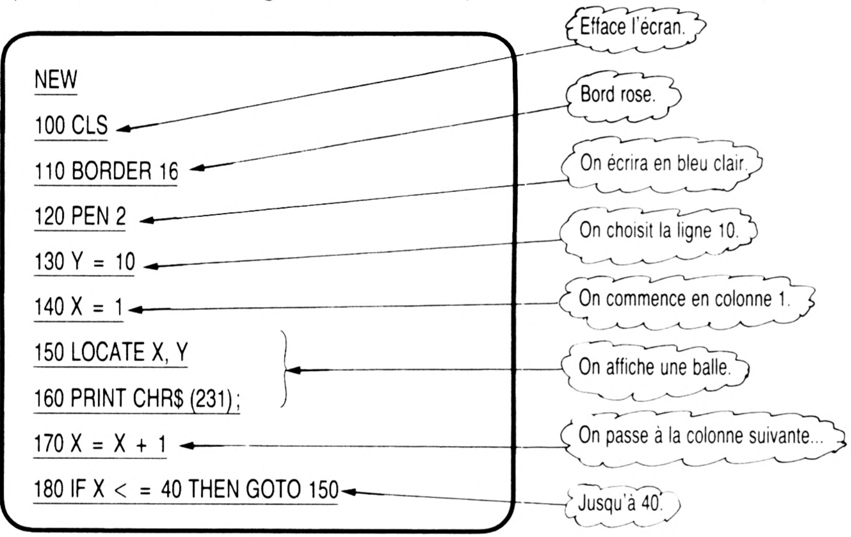
* Utilisez d'autres caractères en modifiant la valeur 231 de la ligne 180.
* Faites tirer le caractère au hasard, en ajoutant :
  175 CA = 128 + INT (128 * RND (1))
  et en modifiant ainsi la ligne 180 :
  180 PRINT CHR$ (CA);

XII. Animer

Nous savons maintenant choisir l'endroit de l'écran où nous voulons afficher quelque chose. Nous pouvons aborder l'étape suivante : faire déplacer un objet sur l'écran.

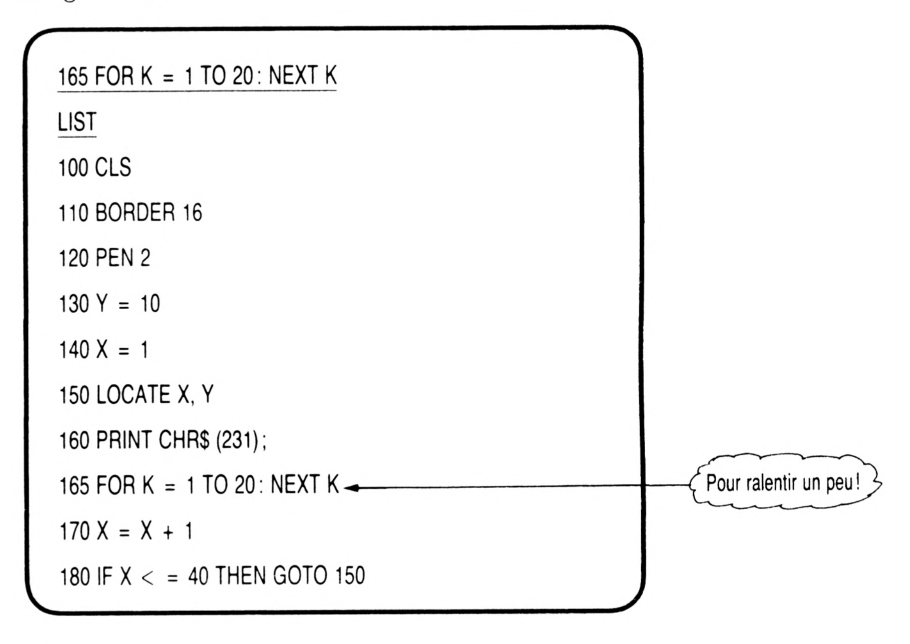
Aligner des balles

Revenez d'abord, si nécessaire, à « l'état initial », puis essayez ce programme qui vous affiche une ligne de « ronds » (ressemblant à des balles).



Peut-être auriez-vous envie d'utiliser des instrutions FOR et NEXT pour faire varier X de 1 à 40. C'est tout à fait possible. Nous l'avons évité ici, car il nous sera ensuite plus facile de « généraliser » notre programme.

Mais, direz-vous! Tout cela n'est pas encore de l'animation. Patience... Nous y venons. Pour l'instant, diminuez un peu la «vitesse d'affichage» en ajoutant la ligne 165:



```
165 FOR K = 1 TO 20: NEXT K
LIST
100 CLS
110 BORDER 16
120 PEN 2
130 Y = 10
140 X = 1
150 LOCATE X, Y
160 PRINT CHR$ (231);
165 FOR K = 1 TO 20: NEXT K
170 X = X + 1
180 IF X < = 40 THEN GOTO 150
```


Vous voyez vos balles s'aligner progressivement devant vos yeux.



Faites-vous la main

* Modifiez à votre gré la position de la «ligne» où apparaissent les balles, en adaptant l'instruction 130.
* Modifiez à votre gré la longueur de la «ligne» en adaptant l'instruction 180.
* Utilisez d'autres caractères que la balle pour former la ligne, en modifiant le code utilisé en 160.
* Modifiez le rythme en adaptant l'instruction 165.

Pour se déplacer horizontalement

Le programme précédent se contente d'afficher le même symbole en différents emplacements. Mais, notre balle ne se déplace pas vraiment sur l'écran. Essayez alors d'ajouter ces deux instructions:

```
167 LOCATE X, Y
168 PRINT CHR$ (32);
LIST
100 CLS
110 BORDER 16
120 PEN 2
130 Y = 10
140 X = 1
150 LOCATE X, Y
160 PRINT CHR$ (231);
165 FOR K = 1 TO 20: NEXT K
167 LOCATE X, Y
168 PRINT CHR$ (32);
170 X = X + 1
180 IF X < = 40 THEN GOTO 150
```

32 est le code du caractère espace. Vous pourriez aussi écrire 168 PRINT " ";

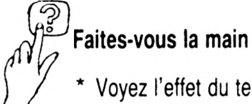
Affichent la balle en colonne X, ligne Y.

Attend un instant.

Affichent un espace à la place de la balle.

Vous voyez qu'après l'affichage de chaque caractère « balle », le programme attend un instant ; il place à nouveau le curseur au même endroit, et il y affiche... un espace (code 32). Cet espace prend la place de la balle et ainsi l'efface.

Finalement, ce programme vous donne *« l'impression de déplacer »* la balle. Voilà comment vous réaliserez de l'animation : vous ne déplacez pas vraiment un objet sur l'écran. Vous le faites simplement apparaître successivement (pendant un instant) en des positions suffisamment rapprochées pour donner l'illusion du mouvement.



Faites-vous la main

* Voyez l'effet du temps d'attente sur la « qualité » de l'animation.

Elle fait des bonds...

Pour l'instant, notre balle se déplace de gauche à droite, d'une colonne à la fois. Modifiez alors ainsi les instructions 140 et 170 :

```
140 X = 1: DX = 1
170 X = X + DX
LIST
100 CLS
110 BORDER 16
120 PEN 2
130 Y = 10
140 X = 1: DX = 1
150 LOCATE X, Y
160 PRINT CHR$ (231);
165 FOR K = 1 TO 20: NEXT K
167 LOCATE X, Y
168 PRINT CHR$ (32);
170 X = X + DX
180 IF X < = 40 THEN GOTO 150
```

Au lieu de X = X + 1.

On introduit une nouvelle variable DX qui précise de « combien » on se déplace à chaque fois.



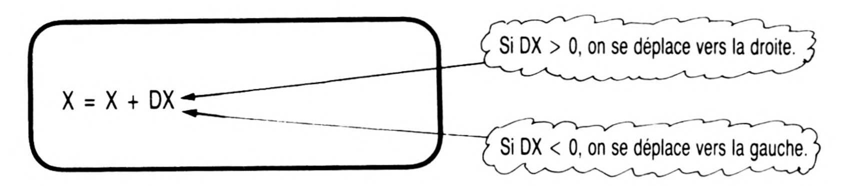

Faites-vous la main

* Essayez ce programme en donnant d'autres valeurs à DX (en ligne 140), par exemple 2, 3,...

* Remplacez 140 par :
  140 X = 40: DX = - 1
  Voyez comme la balle se déplace de droite à gauche de l'écran.
  La façon dont le programme « s'arrête » est par contre peu satisfaisante. Tout ira mieux si vous modifiez ainsi la ligne 180 :
  180 IF X > = 1 THEN GOTO 150

Pour rebondir sur les bords

Vous savez maintenant déplacer la balle de gauche à droite ou de droite à gauche, grâce à l'instruction.



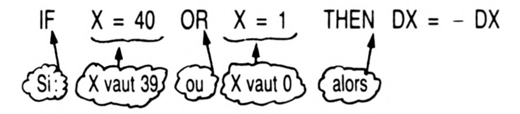
Vous voyez que pour «rebondir», il suffit d'inverser le sens du déplacement. Ainsi, si DX vaut 1, il faudra lui donner la valeur – 1. S'il vaut – 1, il faudra lui donner la valeur + 1. Il existe une instruction qui réalise cela, c'est :

```
DX = - DX
```

Quand faut-il exécuter cette instruction ? Lorsque la balle est au bord, c'est-à-dire quand X vaut 40 (bord droit) *ou* quand X vaut 1 (bord gauche). En Basic, nous pouvons faire cela ainsi.

```
IF X = 40 THEN DX = - DX
IF X = 1 THEN DX = - DX
```

Sur votre Amstrad, vous pouvez condenser ces deux instructions en :



Modifions notre programme :

```
175 IF X = 40 OR X = 1 THEN DX = - DX
180 GOTO 150
LIST
100 CLS
110 BORDER 16
120 PEN 2
130 Y = 10
140 X = 1: DX = 1
150 LOCATE X, Y
160 PRINT CHR$ (231);
165 FOR K = 1 TO 20: NEXT K
167 LOCATE X, Y
168 PRINT CHR$ (32);
170 X = X + DX
175 IF X = 40 OR X = 1 THEN DX = - DX
180 GOTO 150
```

Lorsque l'on atteint l'un des 2 bords, on inverse le sens du déplacement.

Cette fois, si vous n'avez pas fait de fautes de frappe, vous voyez votre balle rebondir sur chaque bord.



Faites-vous la main

* « Sonorisez » votre programme en émettant un son à chaque rebond. Pour cela, modifiez ainsi la ligne 175 :

  175 IF X = 40 OR X = 1 THEN DX = - DX: SOUND 1,60,20

* Voyez ce qui se passe si vous modifiez la valeur 60 de la ligne 175. Essayez, par exemple, 90 puis 40.

* Voyez ce qui se passe lorsque vous modifiez la valeur 20 de cette même ligne 175. Essayez, par exemple, 40, 80, 10 et 5.

* Essayez d'obtenir un déplacement vertical, avec rebonds, en modifiant ainsi les instructions suivantes :

```
140 X = 15: DY = 1
170 Y = Y + DY
175 IF Y = 25 OR Y = 1 THEN DY = - DY
```

Pour déplacer en « oblique »

Jusqu'ici, notre balle se déplaçait, soit horizontalement, par :

```
X = X + DX
```

soit verticalement par :

```
Y = Y + DY
```

Si nous jouons à la fois sur X et Y, nous obtiendrons un déplacement oblique. Pour rebondir, il nous suffit de remarquer que :

— sur les bords verticaux, le rebond se fait par :

```
DX = - DX
```

— sur les bords horizontaux, il se fait par :

```
DY = - DY
```

Voici un programme qui vous permet de choisir la position initiale de la balle et qui la déplace sans cesse sur l'écran.

```
NEW
100 PRINT "DONNEZ COORDONNEES INITIALES"
110 INPUT X, Y
120 CLS
130 BORDER 16
140 PEN 2
150 DX = 1: DY = 1
160 LOCATE X, Y
170 PRINT CHR$ (231);
180 FOR K = 1 TO 20: NEXT K
190 LOCATE X, Y
200 PRINT CHR$ (32);
210 X = X + DX: Y = Y + DY
220 IF X = 40 OR X = 1 THEN DX = - DX: SOUND 1,60,10
230 IF Y = 25 OR Y = 1 THEN DY = - DY: SOUND 1,90,10
240 GOTO 160
```

Pour laisser une trace

Et si vous souhaitez que la balle laisse une « trace » de son passage. Vous avez le choix entre plusieurs possibilités :

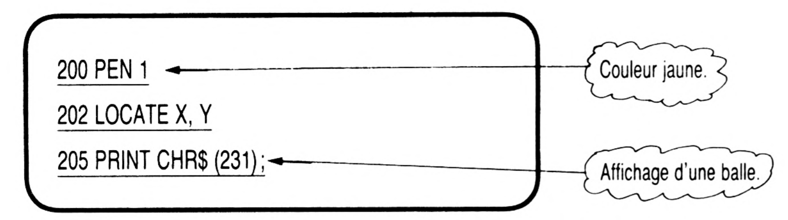
- *ne pas effacer la balle* : il suffit de supprimer les lignes 190 et 200.
- *afficher un caractère autre que espace*, en agissant sur la ligne 200. Par exemple, avec :



```
PRINT "."
```

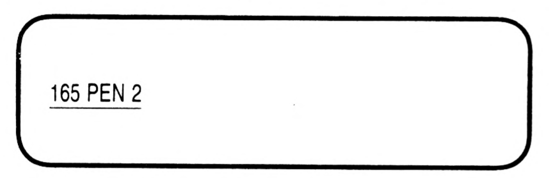
vous obtiendrez une trace formée de points.

- *laisser comme trace une balle de couleur différente*, par exemple bleue. Vous pensez alors probablement à remplacer la ligne 180 par ces trois instructions :



Mais, si vous le faites, vous découvrirez que même la balle mobile est jaune, et non plus bleue. Ceci provient de ce que l'instruction 140 qui fixe la couleur bleue n'est exécutée qu'en début de programme.

Les choses s'arrangent si vous supprimez cette ligne 140 et si vous ajoutez :





Faites-vous la main

* Essayez d'obtenir différents « motifs » sur votre écran à l'aide du programme précédent, modifié pour que la balle laisse une trace jaune. Pour cela, essayez différents emplacements de départ ; modifiez également les limites en jouant sur les instructions 220 (remplacez 40 par d'autres valeurs) et 230 (remplacez 25 par d'autres valeurs).

La balle folle

Jusqu'ici, notre mobile rebondissait sur les bords de l'écran. Mais, dans certains cas, nous pouvons avoir à limiter le déplacement sans pour autant faire un rebond.

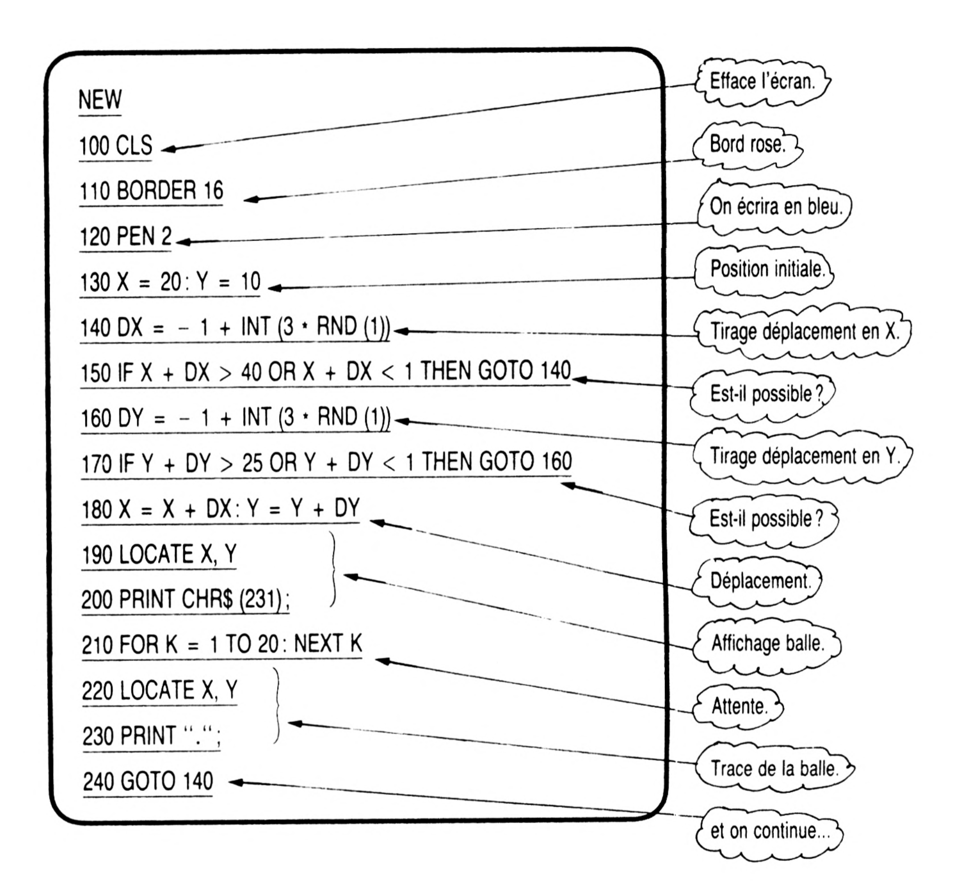
Voici un programme qui déplace au hasard une balle sur l'écran. A chaque fois, nous tirons au hasard une « direction de déplacement » en choisissant pour DX et DY une valeur parmi – 1, 0 ou 1.

Si l'une des quantités :

X + DX
Y + DY

sort des limites de l'écran, nous effectuons un autre tirage.

Nous conservons une trace du déplacement de la balle, en affichant des points.


NEW
100 CLS
110 BORDER 16
120 PEN 2
130 X = 20: Y = 10
140 DX = - 1 + INT (3 * RND (1))
150 IF X + DX > 40 OR X + DX < 1 THEN GOTO 140
160 DY = - 1 + INT (3 * RND (1))
170 IF Y + DY > 25 OR Y + DY < 1 THEN GOTO 160
180 X = X + DX: Y = Y + DY
190 LOCATE X, Y
200 PRINT CHR$ (231);
210 FOR K = 1 TO 20: NEXT K
220 LOCATE X, Y
230 PRINT ".";
240 GOTO 140
Efface l'écran.
Bord rose.
On écrira en bleu.
Position initiale.
Tirage déplacement en X.
Est-il possible ?
Tirage déplacement en Y.
Est-il possible ?
Déplacement.
Affichage balle.
Attente.
Trace de la balle.
et on continue...

XIII. Pour maîtriser vos déplacements :

Dans tous les programmes que nous avons réalisés jusqu'ici, le déplacement des mobiles était choisi par l'ordinateur. Souvent, vous aurez besoin de commander le déplacement d'un objet à l'aide du clavier. Ce sera le cas dans des jeux où vous devrez, par exemple, déplacer une base de tir, piloter une voiture, etc.

Que faut-il savoir faire pour commander un déplacement depuis le clavier ? Il est nécessaire de pouvoir :

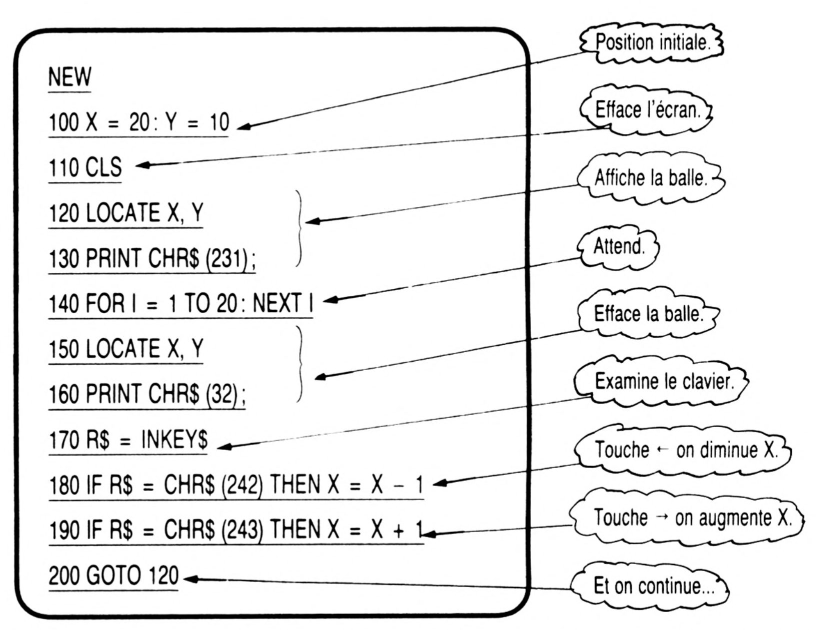
- Déplacer un mobile sur l'écran. C'est ce que nous venons de voir dans le dernier chapitre.
- Examiner le clavier pour savoir si une touche particulière a été enfoncée. Nous avons vu comment procéder dans le chapitre X.

Nous disposons donc des outils de base. Encore faut-il les employer convenablement. C'est ce que nous allons voir maintenant.

Un programme d'entraînement

Essayez ce petit programme qui vous permet de commander le déplacement horizontal d'une balle. Nous conviendrons d'utiliser :

- la touche marquée ← pour le déplacement à gauche. Son code est 242.
- la touche marquée → pour le déplacement à droite. Son code est 243.



Vous constaterez rapidement que cela se termine mal dès que vous laissez la balle aller sur les bords. Vous devez maintenant être habitués à cette situation et comprendre que cela provient de ce qu'aucune « protection » n'est prévue dans le programme.

Pour se protéger

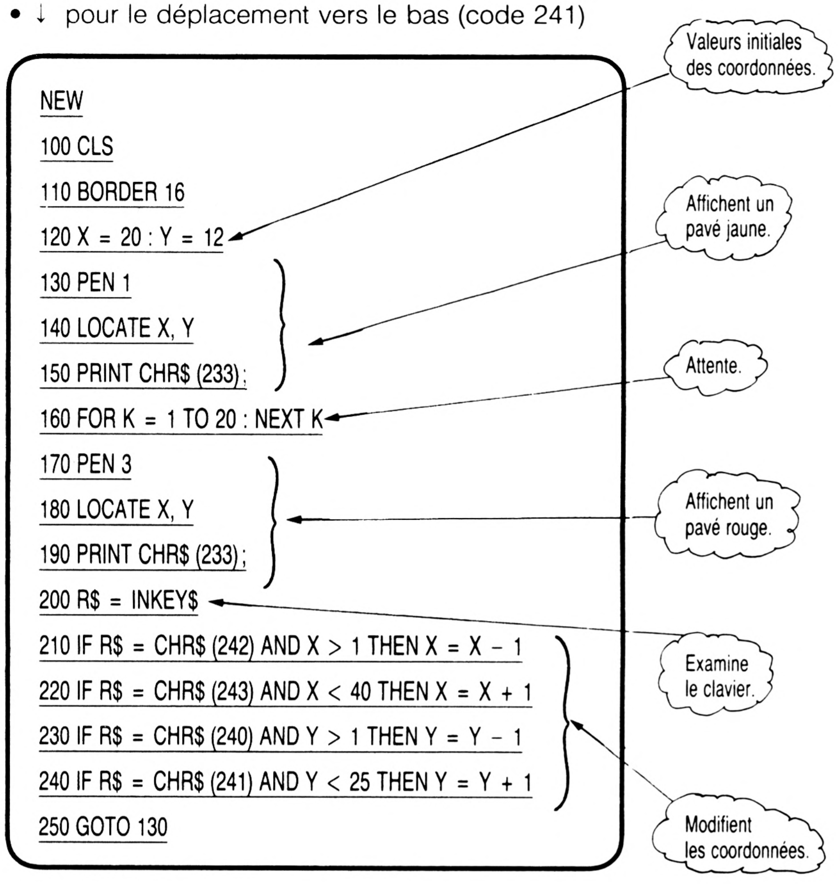
Il nous suffit d'empêcher la valeur de X de devenir inférieure à 1 ou supérieure à 40. Il existe plusieurs solutions. L'une d'entre elles consiste à modifier ainsi les lignes 170 et 180 :

```
180 IF R$ = CHR$ (242) AND X > 1 THEN X = X - 1
190 IF R$ = CHR$ (243) AND X < 40 THEN X = X + 1
```

Pour faire un dessin

Voici un programme qui vous permet de dessiner. Pour cela, vous commandez le déplacement d'un «pavé coloré» à l'aide des touches suivantes :

- ← pour le déplacement vers la gauche (code 242)
- → pour le déplacement vers la droite (code 243)
- ↑ pour le déplacement vers le haut (code 240)
- ↓ pour le déplacement vers le bas (code 241)



XIV. Pour éviter les collisions

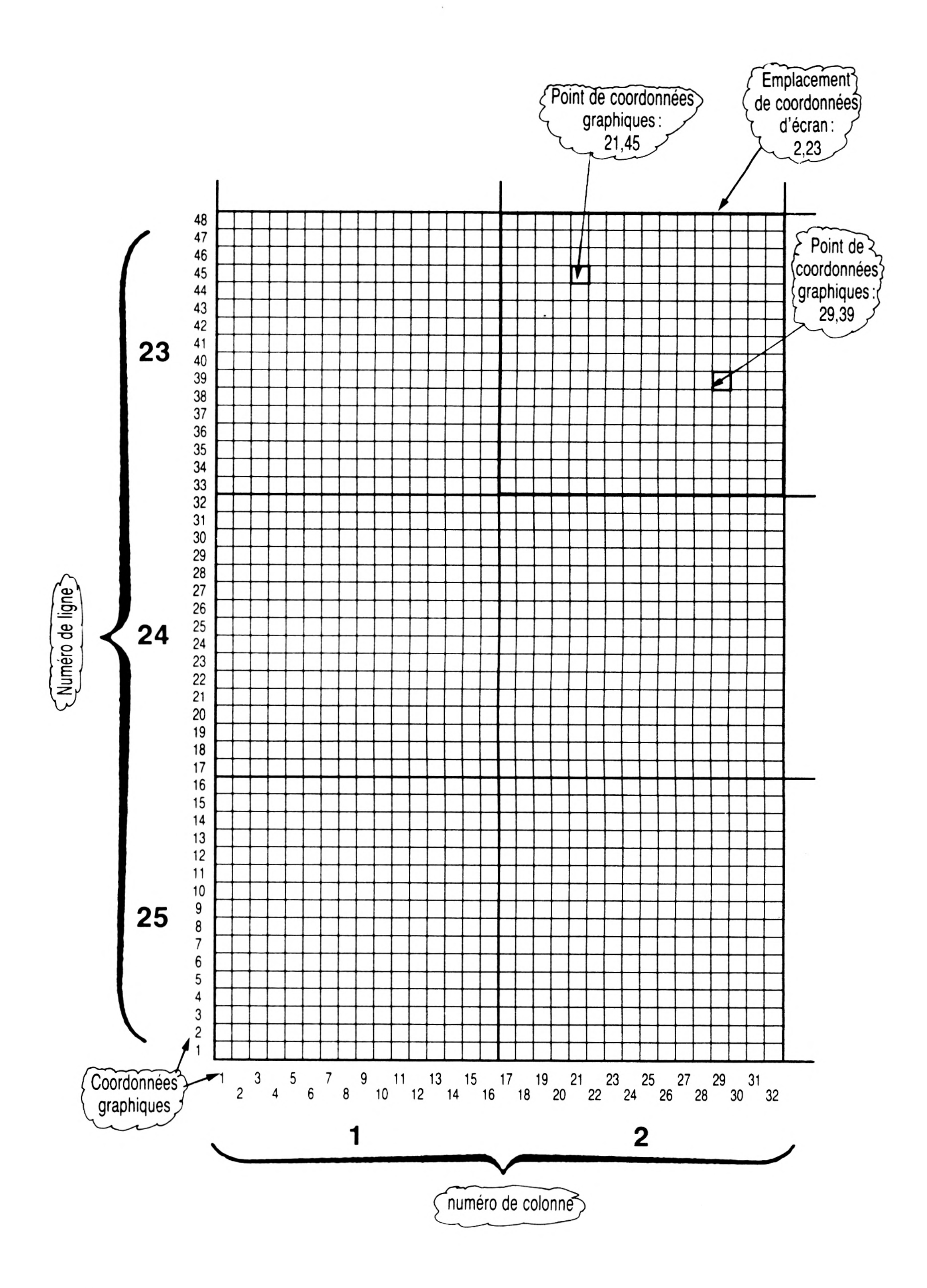
Il vous arrivera souvent d'avoir besoin, dans un programme, de pouvoir connaître ce qui figure en un emplacement de l'écran. Ce sera le cas dans certains jeux ; par exemple :

— pour déplacer une voiture en vous assurant qu'elle ne va pas percuter un mur.
— pour vérifier qu'un missile a atteint sa cible.

Coordonnées « caractère » ou coordonnées graphiques ?

En fait, le Basic de votre Amstrad ne vous permet pas de connaître directement le caractère affiché en un emplacement donné de l'écran. Par contre, il vous permet d'obtenir des indications sur sa couleur.

Une seule difficulté ! Il vous faut employer les « coordonnées graphiques » qui sont différentes des « coordonnées d'écran ». Pour cela, vous devez savoir que chaque emplacement caractère (du mode 1) est en fait subdivisé en 16 × 16 points. Voyez ce dessin qui représente le coin inférieur gauche de l'écran :



Point de coordonnées graphiques : 21,45
Emplacement de coordonnées d'écran : 2,23
Point de coordonnées graphiques : 29,39
23
24
25
Numéro de ligne
Coordonnées graphiques
1
2
numéro de colonne

Les coordonnées graphiques sont ainsi beaucoup plus précises que les coordonnées d'écran. Certaines instructions travaillent avec les coordonnées d'écran ; c'est le cas, par exemple, de LOCATE que nous avons déjà rencontrée. D'autres instructions emploient les coordonnées graphiques. C'est le cas de TEST qui permet de connaître la couleur d'un point ; nous allons l'utiliser dans le jeu que nous vous proposons maintenant.

Le serpent fou

Il s'agit d'un serpent qui se déroule progressivement et sans arrêt sur l'écran. Vous le dirigez à volonté mais vous devez éviter qu'il ne recoupe sa propre trace ou qu'il ne heurte un bord de l'écran.

Pour simplifier la programmation du déplacement, nous allons placer un « cadre » qui représentera le domaine dans lequel le serpent peut se déplacer. Le cadre et le serpent seront représentés par des pavés. Pour vérifier qu'un déplacement est possible, il nous suffira de nous assurer que le nouvel emplacement est disponible ; ce sera vrai si les points graphiques correspondants sont de la couleur du fond (bleu foncé, de code 0). (En fait, on examinera un seul de ces 256 points, bien choisi !)

Pour encadrer

Voyons déjà comment fabriquer le cadre. Il suffit pour cela de tracer un rectangle à l'aide de « pavés ».

```
NEW
100 CLS
110 XD = 2 : XF = 39
120 YD = 2 : YF = 24
130 BORDER 16
140 PEN 2
150 FOR X = XD TO XF
160 LOCATE X, YD
170 PRINT CHR$ (143);
180 LOCATE X, YF
190 PRINT CHR$ (143);
200 NEXT X
210 FOR Y = YD TO YF
220 LOCATE XD, Y
230 PRINT CHR$ (143);
240 LOCATE XF, Y
250 PRINT CHR$ (143);
260 NEXT Y
```

Fixe les limites horizontales du cadre. (110)

Fixe les limites verticales du cadre. (120)

Encre bleue. (140)

Tracent les deux parties horizontales du cadre. (160–190)

Tracent les deux parties verticales du cadre. (220–250)



Faites-vous la main

* Faites varier les dimensions du cadre en modifiant les valeurs de XD, XF, YD et YF. Essayez, par exemple :

```
110 XD = 5 : XF = 25
120 YD = 3 : YF = 20
```

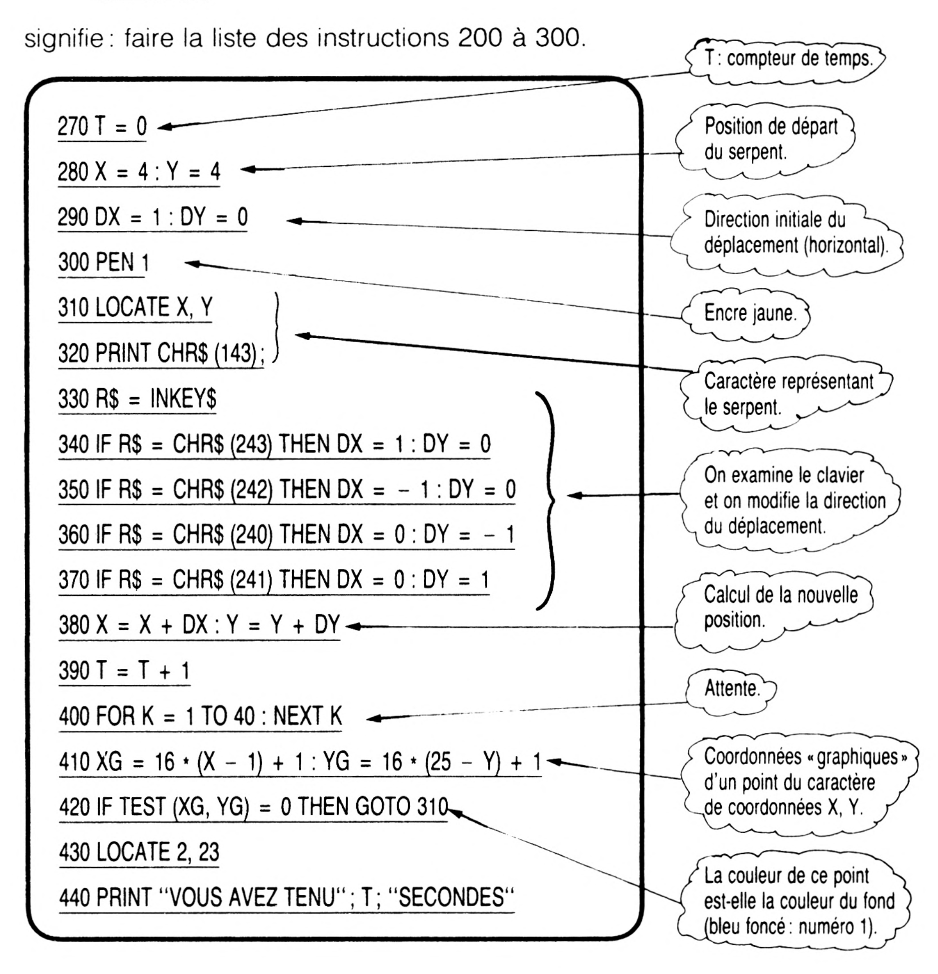
Le jeu

Ajoutez maintenant les instructions suivantes, sans effacer celles que vous venez d'entrer (ne faites surtout pas NEW).

Remarque : vous serez certainement amené à faire une liste. Vous allez découvrir alors que l'ensemble du programme ne tient plus sur l'écran. Seule la fin de la liste sera visible. Sachez que vous pouvez obtenir des *« listes partielles »*. Par exemple :

```
LIST 200-300
```

signifie : faire la liste des instructions 200 à 300.



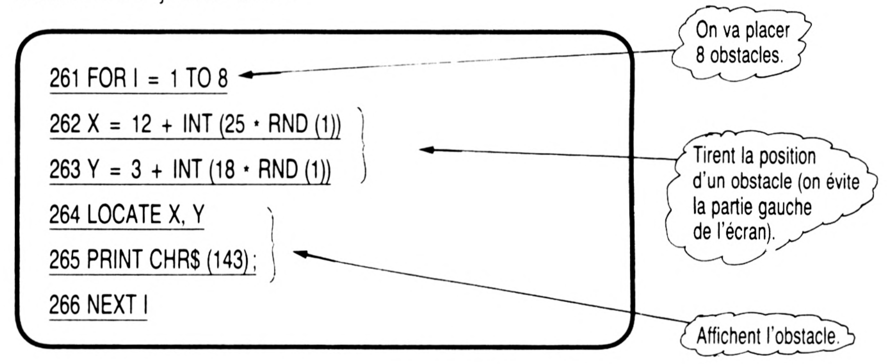


Faites-vous la main

* Voyez comme la durée de la boucle d'attente (en 400) agit sur la difficulté du jeu.

Pour corser le tout

Vous pouvez fabriquer des obstacles, en tirant leur position au hasard. Cette fois, le serpent doit éviter, non seulement les bords et sa propre trace, mais aussi les obstacles. Ajoutez ceci :





Faites-vous la main

* Vous pouvez utiliser d'autres caractères pour représenter les obstacles (et même le cadre ou le serpent). Toutefois, il vous faut savoir que la ligne 410 calcule les coordonnées graphiques d'**un point** particulier de l'emplacement de coordonnées écran X, Y ; il s'agit du point situé dans le **coin inférieur gauche**. Dans le cas d'un pavé, ce point (comme tous les autres) est effectivement coloré. Mais, avec d'autres caractères, ce ne sera pas toujours le cas. Pour vous en convaincre, remplacez le code 143 de la ligne 265 par :

- 249 (les obstacles sont bien « détectés »)
- 225 (les obstacles ne sont plus détectés)
- 238 (ils sont détectés)

Correction des exercices

Chapitre 1

1. Elle écrit : 28

2. PRINT 365 * 21

3. E = 150

4. Elles écrivent : 25

5. Elles écrivent : 100

6. Elles écrivent :
18
15
64
8

7. Vrai.

Chapitre II

1. La première instruction écrit immédiatement : 40.
La seconde sera enregistrée sous le numéro 100.

2. Il faut remplacer l'instruction 100
a) par 100 N = 25
b) par 100 N = 43

3. a) Non. Son exécution écrira : 324.
   b) L'instruction 110 ne sert à rien.
   c) Il suffit de taper :
      130 PRINT B * B

4. a) Nous obtiendrons :
      16
      Syntax error in 120
   b) Il faut la remplacer par :
      120 PRINT N * N
      Cette fois, nous obtiendrons :
      16
      256

5. La ligne 140 comporte une erreur.

6. En frappant simultanément les touches CTRL, SHIFT et ESC.

Chapitre III

1. En utilisant l'une de ces deux instructions :
   PRINT A, B
   PRINT A ; B

2. 25 DIVIZE PAR 5 VOT 5

3. a) 5 DIVISE PAR 4 = 1.25
   b) 5/4 = 1.25
   c) 2/3 = 1.25

4. Il suffit de remplacer la ligne 130 par :
   PRINT "PRODUIT DE" ; A ; "PAR" ; B ; " = " ; C

5. VOICI UN NOMBRE 25
   VOICI SON DOUBLE 75

Chapitre IV

1. Faux.

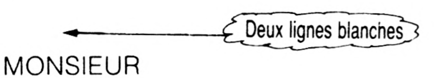
2. Le point d'interrogation montre que Basic attend que vous lui fournissiez une ou plusieurs valeurs.

3. a) MERCI POUR 25
   b) ? Redo from start
   c) ? Redo from start

4. a) MERCI POUR 25
   b) MERCI POUR BONJOUR
   c) MERCI POUR LIST

5. BONJOUR MONSIEUR

6. BONJOUR



MONSIEUR

7. Votre réponse à INPUT est incorrecte. Vous devez en proposer une autre.

Chapitre V

1. IF est un branchement conditionnel
GOTO est un branchement inconditionnel

2. Le premier affiche *sans cesse* :
    SALUT
    MON AMI
Le second affiche *une fois*
    SALUT
puis, il affiche *sans cesse*
    MON AMI

3. I = I + 1

4.



5. a) Dans l'instruction 150, il y a un « mélange » non autorisé de valeurs numériques et chaînes.
b) En ligne 200, on se réfère à une ligne qui n'existe pas.

6. a) En frappant sur la touche



b) en appuyant deux fois sur la touche



Chapitre VI

1.
```
100 PRINT "QUELLE TABLE";
110 INPUT N
120 FOR I = 1 TO 10
130 PRINT I; "FOIS"; N; "="; I * N
140 NEXT I
```

2.
```
100 PRINT "NOMBRE", "CARRE"
110 FOR I = 1 TO 20
120 PRINT I, I * I
130 NEXT I
```

3.
```
100 PRINT "BONJOUR"
110 FOR K = 1 TO 1000: NEXT K
120 PRINT "UN INSTANT SVP"
130 FOR K = 1 TO 5000: NEXT K
140 PRINT CHR$ (141)
150 PRINT "JE M'APPELLE AMSTRAD"
```

4. En ligne 220, Basic a rencontré une instruction NEXT où figure un nom de variable qui ne correspond à aucun compteur mentionné dans une instruction FOR.

Imprimé en France. — JOUVE, 18, rue Saint-Denis, 75001 PARIS
N° 14631. Dépôt légal : Août 1985

Vous débutez en Basic et vous avez choisi un AMSTRAD pour vous initier. Ce livre est fait pour vous ! Quel que soit votre âge, il sera votre premier compagnon de route.

Il ne suppose aucune connaissance préalable. De manière simple et progressive, il vous fera acquérir les bases nécessaires à cette activité passionnante qu'est la programmation.

Grâce aux très nombreuses manipulations qu'il vous propose, il vous conduira à une bonne maîtrise de votre AMSTRAD et de ses particularités graphiques et sonores. Il vous ouvrira la voie de "l'animation".

Très rapidement, vous serez ainsi en mesure de réaliser efficacement vos programmes personnels, y compris, si vous le souhaitez, des jeux vidéo.



8436

couverture: b. lépinois


EYROLLES
Je débute en BASIC
AMSTRAD
C. DELANNOY



AMSTRAD
CPC
OCR
MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA
AC
ME
https://acpc.me/